HUMAN IDENTIFICATION AND TRACKING

by

Melih YÜCE

Submitted to the Institute of Graduate Studies in

Science and Engineering in partial fulfillment of

the requirements for the degree of

Master of Science

in

Electrical and Electronics Engineering

Yeditepe University

2011

HUMAN IDENTIFICATION AND TRACKING

APPROVED BY:

Assoc. Prof. Dr. Cem Ünsalan (Advisor)	........[illegible]........
Assist. Prof. Dr. Duygun E. Barkana	........[illegible]........
Assist. Prof. Dr. Dionysis Goularas	........[illegible]........

DATE OF APPROVAL: …./…./ 2011

ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis advisor Assoc. Prof. Dr. Cem Ünsalan for his assistance, encouragement and advises along this thesis study. Research assistant Rıfat Benveniste has also deserves thanks due to his help to my project. Moreover, I am also very happy to be a part of Yeditepe University. I am so thankful to the instructors of my courses that I took during my Master of Science.

I would also thank to my family members for their encouragement and financial support that they have given me during this study.

Finally, I would like to dedicate this thesis to my family and my wife Ezgi Yüce.

ABSTRACT

HUMAN IDENTIFICATION AND TRACKING

In recent years, face detection and recognition have been very popular in both scientific research communities and the market. But it still remains very challenging in real–time applications. The machine learning and computer graphics communities are also increasingly involved in face recognition. This common interest among researchers working in diverse fields is motivated by the remarkable ability of recent algorithms to recognize people and the fact that human activity is the primary concern both in everyday life and in cyberspace. Face tracking and recognition is also very popular in security systems. The ability of these methods help developing reliable human surveillance systems. Both tracking and recognition systems can be used widely by the police and private security firms. In this MSc. thesis study, we implement a human surveillance system using face detection and recognition as a subpart of a human surveillance system.

ÖZET

İNSAN TANIMA VE İZLEME

Yüz bulma ve tanıma sistemleri geçtiğimiz yıllarda bilimsel araştırma grupları ve piyasa tarafından oldukça geniş ilgi görmüştür. Halen bu sistemlerin gerçek zamanlı uygulamalarının geliştirilmesi oldukça zorlayıcı bir problem olarak görülmektedir. Ayrıca makine öğrenmesi ve bilgisayarda grafik işleme toplulukları da yüz bulma ve tanıma çalışmalarına dahil olmuşlardır. Farklı branşlarda uzmanlaşmış bu araştırmacıların ortak ilgisi, yüz bulma ve tanıma problemlerinin çözümünün gündelik ve sanal hayata kazandıracağı artı değerin yadsınamayacak bir öneme sahip olmasından kaynaklanmaktadır. Yüz takibi ve tanıma sistemleri güvenlik sistemlerininde de çok popülerdir. Bu sistemlerin kabiliyeti insan izleme sistemlerinin geliştirilmesinde çok etkilidir. Bu takip ve tanıma sistemleri Emniyet Müdürlüğü ve özel güvenlik firmaları tarafından kullanılmaya çok müsaittir. İnsan izleme sistemlerinin günümüzdeki öneminden dolayı, bu yüksek lisans tezinde yüz bulma ve tanıma metotlarını kullanarak insan takibi yapan bir sistem geliştirdik.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
TABLE OF CONTENTS vi
LIST OF FIGURES viii
LIST OF SYMBOLS / ABBREVIATIONS x
1. INTRODUCTION 1
2. LITERATURE REVIEW 2
2.1. FACE DETECTION 2
2.1.1. Knowledge-based Methods 2
2.1.2. Feature-based Methods 3
2.1.2.1. Facial Features 3
2.1.2.2. Skin Color 4
2.1.2.3. Texture 5
2.1.2.4. Multiple Features 5
2.1.3. Template Matching Methods 5
2.1.4. Appearance-based Methods 6
2.1.4.1. Neural Networks 6
2.1.4.2. Support Vector Machines (SVM) 6
2.1.4.3. Principal Component Analysis (PCA) 7
2.1.4.4. Distribution-based method 7
2.2. FACE RECOGNITION 7
2.2.1. Appearance-based (view-based) Method 9
2.2.1.1. PCA and Kernel PCA 9
2.2.1.2. 3D Morphable Model 17
2.2.2. Feature-based Method 18
2.2.2.1. Feature-based Elastic Bunch Graph Matching (EBGM) 18
2.2.2.2. Hidden Markov Model (HMM) 18
2.2.3. Combined Method 19

2.3. TRACKING 19
2.3.1. Kernel Tracking 21
2.3.1.1. Template and density-based appearance models 22
2.3.1.2. Multiview appearance models 26
2.3.2. Silhouette Tracking 26
2.3.2.1. Shape matching 28
2.3.2.2. Contour tracking 28
2.3.3. Point Tracking 29
2.3.3.1. Deterministic methods for correspondence 30
2.3.3.2. Statistical methods for correspondence 32
3. THE PROPOSED SYSTEM 34
3.1. FACE DETECTION 35
3.1.1. Adaboost Algorithm 35
3.1.2. Skin Detection 39
3.2. TRACKING 45
3.3. FACE RECOGNITION 47
3.4. THE OVERALL SYSTEM 51
3.4.1. Initialization of The System 53
3.4.2. Execution of The System 53
3.4.2.1. Tracking with two cameras 54
3.4.2.2. Tracking with four cameras 56
4. EXPERIMENTAL RESULTS 61
4.1. EXPERIMENTS WITH TWO CAMERAS SETUP 61
4.2. EXPERIMENTS WITH FOUR CAMERAS SETUP 61
4.3. EXPERIMENTS ON FACE RECOGNITION 63
4.3. LIMITATIONS OF THE PROPOSED SYSTEMS 65
5. CONCLUSIONS 67
APPENDIX A: INITIALIZATION FOR RECOGNITION 68
APPENDIX B: BACKGROUND EXTRACTION 70
APPENDIX C: FUNCTIONS 73
APPENDIX D: MAIN FUNCTIONS 77
REFERENCES 81

LIST OF FIGURES

Figure 2.1. Examples of the Viola and Jones features. 4

Figure 2.2. The main steps of face recognition and its application areas. 8

Figure 2.3. a. Example images from the ORL database, b. Mean face obtained from the ORL database 12

Figure 2.4. Some examples of the eigenfaces, sorted with respect to decreasing eigenvalues. 13

Figure 2.5. The cumulative sum curve for the eigenvalues 14

Figure 2.6. A typical eigenvalue spectrum 14

Figure 2.7. Elastic Bunch Graph Matching Method 18

Figure 2.8. a. Rectangle kernel tracking, b. c. silhouette tracking, d. point tracking 21

Figure 2.9. Mean-shift tracking iterations 23

Figure 2.10. Results of the robust online tracking method 24

Figure 2.11. Tracking features using the KLT tracker. 25

Figure 2.12. a. The measurements consist of image edges computed in the normal direction to the contour b. Level set contour representation 27

Figure 2.13. Contour tracking results 29

Figure 2.14. Point correspondence 31

Figure 2.15. Different motion constraints 31

Figure 3.1. Camera settlement for the first scenario 34

Figure 3.2. Camera settlement for the second scenario 35

Figure 3.3. The value of the integral image at point (x,y) 36

Figure 3.4. The sum of the pixels within rectangle D 37

Figure 3.5. The processed image after the Adaboost algorithm 39

Figure 3.6. The CbCr plane at constant luminance Y=0.5 40

Figure 3.7. RGB color image and its components 41

Figure 3.8. YCbCr color image and its components 42

Figure 3.9. Histogram of skin color in Cr component of the YCbCr image 43

Figure 3.10. Face area after thresholding 43

Figure 3.11. Final state of face image 44

Figure 3.12. The skin detection result 44

Figure 3.13. Flow diagram of the first scenario 45

Figure 3.14. Flow diagram of second scenario 46

Figure 3.15. Database images gathered from Face Recognition Homepage 47

Figure 3.16. Flow diagram of the system 52

Figure 3.17. Displays of two cameras A and B 54

Figure 3.18. Flow diagram of tracking algorithm for two cameras 55

Figure 3.19. Displays of four cameras A, B, C and D 57

Figure 3.20. The decision for the recognition in four camera system 58

Figure 3.21. Flow Diagram of tracking algorithm for two cameras 60

Figure 4.1. Working system with two cameras 61

Figure 4.2. Working system with four cameras 62

Figure 4.3. Face recognition results of our algorithm using the database in Face Recognition Homepage 63

Figure 4.4. Database images used in the system 64

Figure 4.5. Result after the recognition process 65

LIST OF SYMBOLS / ABBREVIATIONS

A	Difference Matrix
I	Image
i(x,y)	Original image
ii(x,y)	Integral image
M_t	Number of the training images
N_x	Image size in x coordinates
N_y	Image size in y coordinates
P	Dimension of the image space
s(x,y)	The cumulative row sum
v	Eigenvector
w	Weight
X	Covariance Matrix
δ	Similarity
ε	Distance Measure
Θ	Distance Threshold
μ	Eigenvalue
Γ	Image vector
Φ	Mean subtracted image
Ψ	Mean face
Ω	Weight matrix
ω	Projection
1D	One Dimension
2D	Two Dimensions
3D	Three Dimensions
Adaboost	Adaptive Boosting
CCD	Charge Coupled Device
EBGM	Elastic Bunch Graph Matching

HCI	Human Computer Interaction
HMM	Hidden Markov Model
HSV	Hue, Saturation, Value color triple
KLT	Kanade-Lucas-Tomassi
ORL	Operations Research Laboratory
PCA	Principle Component Analysis
RGB	Red, Green, Blue color triple
SVM	Support Vector Machine
TV	Television
WKPCA	Weighted Kernel Principal Component Analysis
YCrCb	Luma, Red difference, Blue difference color triple
YES	Luma and chrominance color triple
YIQ	Luma and chrominance color triple

1. INTRODUCTION

In recent years, advances in image processing techniques and the decreasing cost of various image and video acquisition devices encouraged the development of many applications. Among these, human face detection and tracking is the most attractive one for researchers since they have a wide range of applications in the areas of surveillance, security systems, human computer interaction, animation, and biometrics.

In this thesis study, we develop a system to detect, track, and recognize human faces. There are many approaches to overcome to the problem of face detection and tracking. In terms of face detection, previous studies can be classified into four main groups. In the literature review chapter, we will introduce them shortly. Then, we will focus on the Adaboost and skin color based face detection method, since we used them in our system. We also consider the well-known tracking methods in the literature review chapter. Among these, we pick the point tracking method for face tracking. In the following chapters, we explain this method in detail. Face recognition is also very important for the human surveillance systems. Therefore, we added a subpart to our system to recognize faces. As in previous methods, we explain what has been done one face recognition in the literature review chapter of the thesis. In implementation, we benefit from the Principle Component Analysis (PCA) based face recognition method. We will explain its working principles in the following chapters. In implementing our face detection, tracking and recognition system, we consider two strategies. In the first one, we have two cameras. In the second strategy, we have four cameras in implementation.

The layout of the thesis is as follows. We will first start to look the literature to review which algorithms are valid for face detection, tracking, and recognition in Chapter 2. Then, we explain the implementation of our algorithms one by one and the whole system in Chapter 3. Chapter 4 comprises our experiments and results.

2. LITERATURE REVIEW

In this chapter, we review the existing work on face detection, recognition and tracking. For ease of understanding, we grouped these studies as follows.

2.1. FACE DETECTION

The definition of the face detection problem is finding or extracting one or more human faces from a given image. According to users' expectations, different methods for face detection have been proposed in the literature. Ideally, a robust and good face detector algorithm accurately extracts all faces in images regardless of their position, pose, scale, color, orientation, shape, and external lighting conditions. In addition to accurately finding faces, another significant problem is the detection time. This is very important because face detection should be done very fast in tracking, surveillance applications, and video streaming phones. Real-time working requirement prohibited many algorithms that precisely extract faces using extensive computations. Existing techniques for face detection can be classified into four main categories as: knowledge-based, feature-based, template matching based and appearance-based methods.

2.1.1. Knowledge-based Methods

Face detection in this group is examined on the rules derived from the researcher's knowledge on human faces. In general, these rules belong to top-down (image-based) approach. Top-down approach means that face regions are determined without resorting to the identification of individual facial components such as lips, noses and eyes. So, the basic rules are as: the color density has the maximum value in the center of human face; there is a large illumination difference between human face region and the non-face region; all facial features like eyes, lips and nose are symmetric.

Two variants for this method have been proposed. The first one is the vertical and horizontal projection diagram method. Diagrams are extracted and decisions are made if the areas with high projection rate include lips, nose and eyes [1]. The second method focuses on the sequential resolutions approach. In this method, the image is processed first in low resolution and the face center is located. After this process, in a finer resolution histogram equalization and edge detection is applied. Finally, based on this face center, the image is processed in high resolution to search the lip, nose and eyes [2].

Advantages of the knowledge-based methods are as follows. Finding face properties and the rules which describe the relations between them are very easy. Extracting face properties from an image based on these rules are also very simple. These methods are successful on images that do not have cluttered backgrounds. Disadvantages of these methods are using the rules of human knowledge including small details. These may give incorrect results and faces in different poses may not be detected.

2.1.2. Feature-based Methods

In contrast to the knowledge-based top-down method, researchers have been trying to find invariant features for face detection, like edge color variety, color density, design and shape. Feature-based methods are also called bottom-up approaches. Bottom-up approach means that the precise face regions are constructed from the individual facial components. In other words, the facial components should be extracted prior to the determination of the precise location of face regions in the image. Feature-based methods are classified into four categories as: facial features, skin color, texture and multiple features.

2.1.2.1. Facial Features

Methods of facial features are interested in eyes, nose, lips, eyebrow positions and presence of them and their orientations. Sirohey [3] proposed a localization method to segment faces from a cluttered environment for face identification. This method depends on edge map extraction, removing and grouping edges so that only the ones on the face counter are preserved. Graf [4] has developed a method to locate facial features and faces in gray scale images. The method uses band-pass filtering and morphological operations to enhance regions with high intensity values and certain shapes such as eyes. There are other methods to identify facial components which use Gaussian filters and its derivatives and Gaussian distributions in arrangement of facial components with mutual distances [5, 6, 7].

In the paper of Viola and Jones [8], three types of Haar-like features are utilized; namely, two, three, and four rectangle features (see Figure 2.1), which help in computing the difference between the sum of pixels within rectangular areas in the image. These Haar-like features are considered as face features like lips, eyes, eyebrow etc. We use this algorithm to detect faces. Therefore, we will explain it in detail in the next chapter.

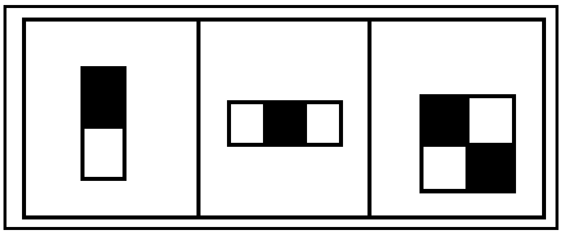

Figure 2.1. Examples of the Viola and Jones features.

2.1.2.2. Skin Color

From face detection to hand tracking, skin color representation is an effective feature. Although different people have different skin colors like fair-skinned, dark-skinned etc., many proposed studies have shown that the main difference is their intensity not their chrominance. Face detection in color images is used with many proposed color spaces to label skin pixels in the image. Some of the color spaces can be given as RGB, normalized RGB, YCbCr, HSV, YIQ, and YES [9 - 17]. An image is formed of many pixels. Each pixel in an image is represented with its intensity value of 0 to any values like 63; 127; and 255. These color spaces represent a pixel with three features. RGB color space represents a pixel with red, green and blue components. YCbCr color space represents a pixel with luma, blue-difference and red-difference components. HSV color space represents a pixel with hue, saturation and value components. Many methods have been developed for skin color modeling. The simplest model is to define a region of skin tone pixels including Cb and Cr values. With carefully chosen thresholds values, a pixel can be classified to have skin tone if it is falls into the chosen threshold values. We will explain how to use this method in the next chapter.

2.1.2.3. Texture

Human faces have a distinct feature that can be separated from other objects in the image. A method is developed by Augusteijn and Skufca [18] that infers the presence of a face through the identification of face-like features. Three types of features are considered as texture in this method: skin, hair and others.

2.1.2.4. Multiple Features

Numerous methods that combine several facial features have been proposed to locate or detect faces. Most of them utilize global features such as size, skin color and shape to find face candidates, and then verify these candidates using local and detailed features such as eyes, nose, brows, and hair. A typical approach begins with the detection of skin-like regions. Next, skin-like pixels are grouped together using connected component analysis or clustering algorithms. If the shape of a connected region has an elliptic or oval shape, it becomes a face candidate. Finally, local features are used for verification [10, 19, 20].

Advantage of feature-based method is independency of characteristics of human faces for different illumination conditions and poses. The disadvantage of this method is that, the process of detecting faces is hard under different illumination and complex environments.

2.1.3. Template Matching Methods

These methods use templates that are defined beforehand. Various standard patterns are stored in a database to describe the face as a whole or the facial features separately. The image is checked if there is a face in the image that matches with the template in database [21 - 23]. The correlations between the input image and the stored templates are computed for detection. The existence of a face is determined based on these correlation values. This approach has the advantage of being simple to implement and easy for calculations. Disadvantage of this method is its being inadequate for face detection since it cannot effectively deal with variation in scale, pose, and shape.

2.1.4. Appearance-based Methods

In contrast to template matching methods where templates are predefined, the templates in appearance-based methods are learned from examples in images. In general, appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face and non-face images. These methods can benefit from examples and have been demonstrated to be more robust and useful than other methods [24].

Generally all kinds of appearance-based methods are actualized with vertically and horizontally scanning test images. To improve the success rate of this algorithm, sub sampling is done by decreasing the height and width of the image. Sub sampling and scanning continues until the image become as small as possible.

Many different classifiers and learning methods are available to perform appearance-based detection. These methods include artificial neural networks, principal component analysis, support vector machines and distribution-based methods [25 - 31].

2.1.4.1. Neural Networks

This technique uses light correction and histogram equalization methods to standardize the image due to training with complex facial patterns. Besides, the implementation of this approach is simple. This method examines small windows of an image, and decides whether each window contains a face. Therefore, it cannot effectively deal with variation in scale, pose and shape. It has a straightforward procedure for aligning positive face examples for training. With this technique, simple heuristics (such as faces rarely overlap in images) can further improve the accuracy. The advantage of using neural networks in face detection is the feasibility of training a system to capture the complex class conditional density of face patterns.

2.1.4.2. Support Vector Machines (SVM)

Support Vector Machines can be considered as a new paradigm to train a polynomial function, neural networks classifiers. While most methods for training a classifier (like Bayesian and neural networks), are based on of minimizing the training error, SVM operates on another induction principle, called structural risk minimization, which aims to minimize an upper bound on the expected generalization error. An SVM classifier is a linear classifier where the separating hyperplane is chosen to minimize the expected classification error of the unseen test patterns. This optimal hyperplane is defined by a weighted combination of a small subset of the training vectors, called support vectors.

2.1.4.3. Principal Component Analysis (PCA)

This aim of this method is the selection of a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values for all features. PCA is useful for finding more informative, uncorrelated features in the image. We will cover this method in the face recognition chapter in detail.

2.1.4.4. Distribution-based method

This technique uses histogram equalization and masking procedures. With the help of histogram equalization technique, we can eliminate the imaging effects due to illumination variations and different camera input gains in the image. Masking provides reducing the unwanted background noise in face template images. Moreover, the best fit brightness plane is found and subtracted from the image. This way, heavy shadows that are caused by extreme lighting are reduced.

Advantages for appearance-based methods can be given as follows. They have successful algorithms, successful proved results. They are fast and their processing procedures are effective. They work well in different face poses. As disadvantages concerned, we can only give the fact that these algorithms should have a lot of positive and negative example images in order to run well.

2.2. FACE RECOGNITION

Face detection is the first step for face tracking or recognition. Many commercial face recognition applications are available such as person identification, security systems, criminal investigations, image and film processing. The purpose here is to find the best match between the given face image and the face in the sequence of images captured by camera. The face recognition system should be able to identify or verify one or more persons in the scene using a pre-defined image database.

As explained above, before performing face recognition, the system should determine whether or not there is a face in the given image. Once the face region is detected, it should be isolated from the scene for face recognition. Features of the face obtained from the face detection system are used in the face recognition process. Therefore, the feature extraction step is placed between face detection and recognition. However, face detection and recognition are often performed simultaneously. The overall process is shown in Figure 2.2.

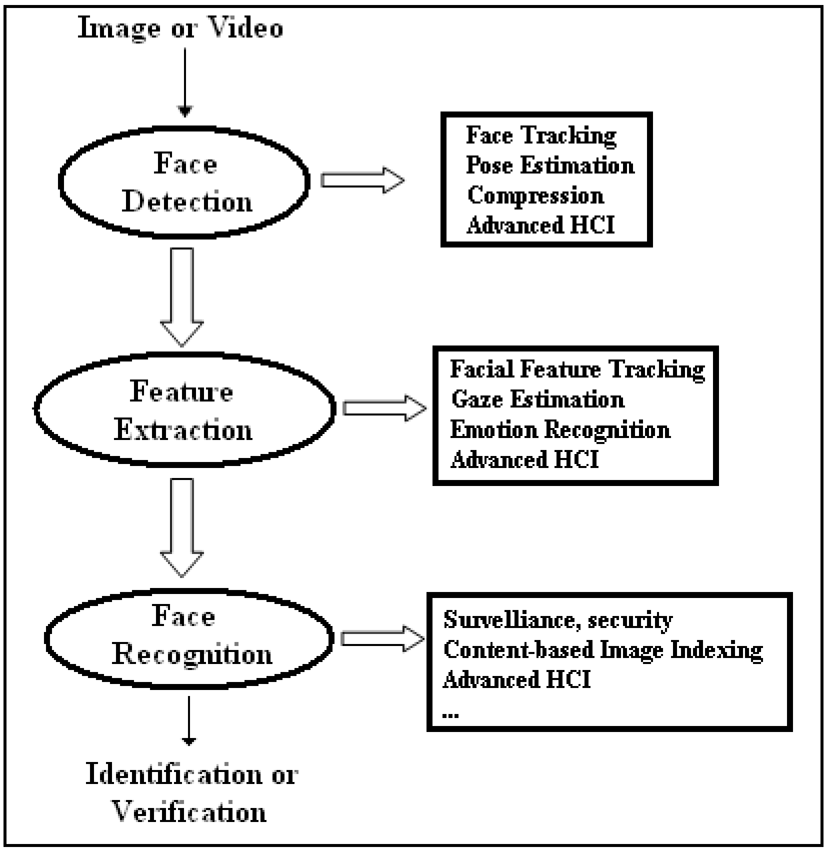

Figure 2.2. The main steps of face recognition and its application areas.

Face recognition is complicated if the images of the face is taken in different poses with different angles [32]. To handle these problems several methods are proposed such as: image resizing, (to align faces and eyes on the same line for all images); background subtraction, (to delete image background and human hair from the image); histogram equalization, (to gain best color contrast distribution); face normalization with zero mean. Face recognition algorithms are classified into three main groups as: appearance-based (view-based), feature-based, and combined.

2.2.1. Appearance-based (view-based) Method

These methods use specifications of the face such as the shape or the area of the face, distances between eyes and mouth etc.

2.2.1.1. PCA and Kernel PCA

An image can be represented as a space having dimensions equal to the number of pixels forming up the image and having values in 32 the range of the pixels values. Thus, for example for a gray scale image of size (N_x x N_y), the dimension of the image space is P= N_x x N_y. For the case of gray scale images, in each dimension, the image can have an intensity value between 0 and 255. An image can be thought as a point in the image space by converting it to a long vector by arranging each column of the image one after the other.

After all the face images are converted into vectors, they will cluster at a certain location in the image space as they have similar structure, having eye, nose and mouth in common and their relative position correlated. This correlation is the main start point for the eigenface analysis. The eigenface method tries to find a lower dimensional space for the representation of the face images by eliminating the variance due to non-face images; that is, it tries to focus on the variation just coming out of the variation between the face images. The eigenface method is the implementation of PCA over images. In this method, the features of the studied images are obtained by looking for the maximum deviation of each image from the mean image. This variance is obtained by getting the eigenvectors of the covariance matrix of all images. The eigenface space is obtained by applying the eigenface method to the training images. Later, the training images are projected into the eigenface space. Next, the test image is projected into this new space and the distance of the projected test image to the training images is used to classify the test image. In the standard eigenface procedure suggested by Turk and Pentland [28], the nearest mean classifier is used for the classification of test images.

In mathematical terms, let I be the image matrix with (N_x x N_y) pixels. This image is converted to the image vector Γ of size (P x 1) where P = (N_x x N_y); that is the image matrix is reconstructed by adding each column one after the other as

$$\Gamma = [\Gamma_1 \ \Gamma_2 \ ... \ \Gamma_{Mt}] \tag{2.1}$$

is the training set of image vectors and its size is (P x M_t) where M_t is the number of the training images. Based on these, we can form the arithmetic average of the training image vectors at each pixel point as

$$\Psi = \frac{1}{M_t} \sum_{i=1}^{M_t} \Gamma_i \tag{2.2}$$

This arithmetic image can be used to obtain the difference of the training image from the mean image as

$$\Phi = \Gamma - \Psi \tag{2.3}$$

For all images, we can from the matrix of all the mean subtracted training image vectors as

$$A = [\Phi_1 \ \Phi_2 \ ... \ \Phi_{Mt}] \tag{2.4}$$

Using A, we can form the covariance matrix of the training image vectors as

$$X = A \cdot A^T = \frac{1}{M_t} \sum_{i=1}^{M_t} \Phi_i \Phi_i^T \tag{2.5}$$

An important property of the eigenface method is obtaining the eigenvectors of the covariance matrix, X. For a face image of size (N_x x N_y) pixels, the covariance matrix should be of size (P x P), P being (N_x x N_y). This covariance matrix is very hard to work with due to its huge dimension causing computational complexity. On the other hand, the eigenface method calculates the eigenvectors of the (M_t x M_t) matrix, M_t being the number of face images, and obtains (P x P) matrix using the eigenvectors of the (M_t x M_t) matrix as follows. Initially, a matrix Y is defined as,

$$\mathrm{Y} = \mathrm{A}^T \cdot \mathrm{A} = \frac{1}{M_t} \sum_{i=1}^{M_t} \Gamma_i^T \Gamma_i \tag{2.6}$$

which is of size (M_t x M_t). Then, the eigenvectors v_i and the eigenvalues μ_i of Y are obtained as

$$\mathrm{Y} \cdot v_i = \mu_i \cdot v_i \tag{2.7}$$

The value of Y is put in this equation,

$$\mathrm{A}^{\mathrm{T}} \cdot \mathrm{A} \cdot v_i = \mu_i \cdot v \tag{2.8}$$

Both sides are multiplied by A from left,

$$\mathrm{A} \cdot \mathrm{A}^{\mathrm{T}} \cdot \mathrm{A} \cdot v_i = \mathrm{A} \cdot \mu_i \cdot v_i \tag{2.9}$$

The necessary matrix arrangements are made,

$$\mathrm{A} \cdot \mathrm{A}^{\mathrm{T}} \cdot \mathrm{A} \cdot v_i = \mu_i \cdot \mathrm{A} \cdot v_i \tag{2.10}$$

As μ_i is a scalar, this arrangement can be done as

$$\mathrm{X} \cdot \mathrm{A} \cdot v_i = \mu_i \cdot \mathrm{A} \cdot v_i \tag{2.11}$$

Now, we can group $\mathrm{A} \cdot v_i$ and call a variable $\upsilon_i = \mathrm{A} \cdot v_i$. It is easy to see that

$$\upsilon_i = \mathrm{A} \cdot v_i \tag{2.12}$$

is one of the eigenvectors of $X = A \cdot A^T$ and its size is (P x 1).

Thus, it is possible to obtain the eigenvectors of X by using the eigenvectors of Y. A matrix of size (M_t x M_t) is utilized instead of a matrix of size (P x P) (i.e. [{N_x x N_y} x {N_x x N_y}]). This formulation brings substantial computational efficiency. In Figure 2.3, some example images and mean image of the images from the ORL database are given. In Figure 2.4, some characteristic eigenfaces obtained from this database can be seen. The eigenfaces are in fact (P x 1) vectors for the computations; in order to see what they look like, they are rearranged as (N_x x N_y) matrices.

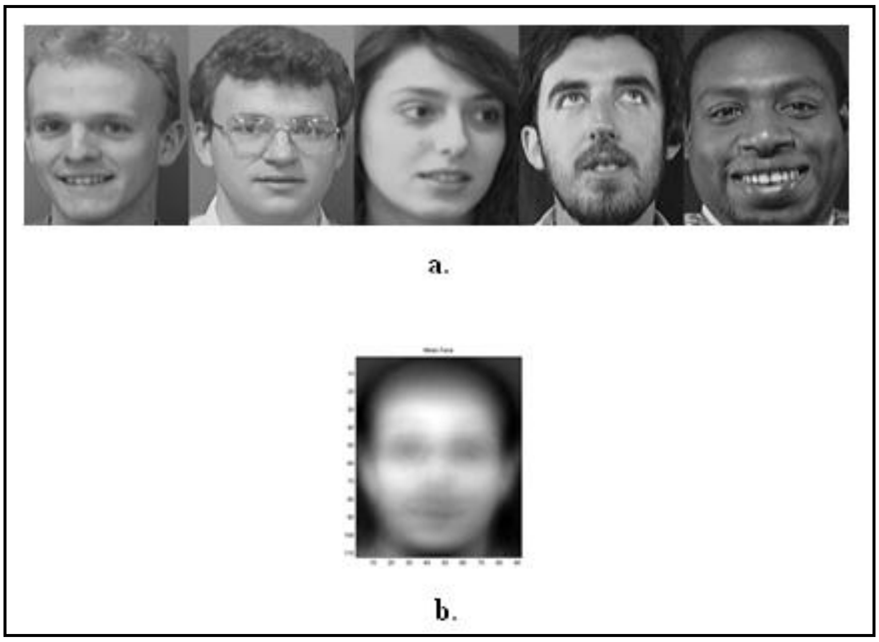

Figure 2.3. a. Example images from the ORL database, b. Mean face obtained from the ORL database

Instead of using M_t of the eigenfaces, $M' \leq M_t$ of the eigenfaces can be used for the eigenface projection. This is achieved to eliminate some of the eigenvectors with small eigenvalues, which contribute less variance in the data. In Figure 2.5, the cumulative sum of the eigenvalues, and in Figure 2.6 a typical eigenvalue spectrum can be observed.

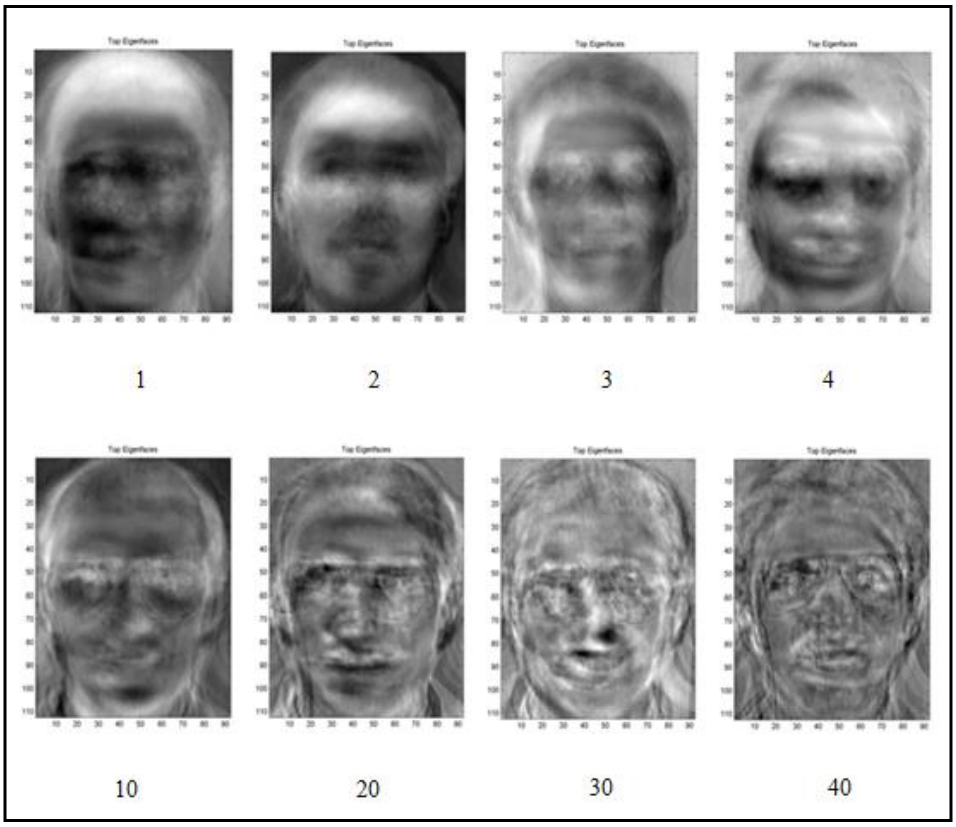

Figure 2.4. Some examples of the eigenfaces, sorted with respect to decreasing eigenvalues.

From Figures 2.5 and 2.6, it can be easily observed that most of the generalization power is contained in the first few eigenvectors. For example, 40% of the total eigenvectors have 85 – 90% of the total generalization power. Thus, using 40% of the total number of eigenvectors may end up with reasonable classification results.

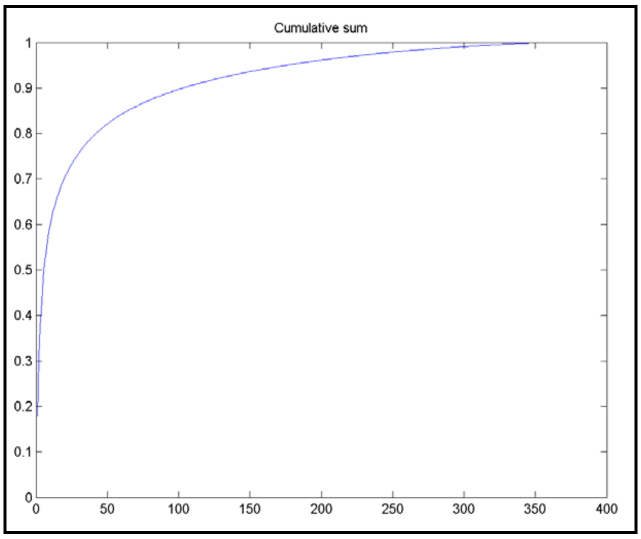

Figure 2.5. The cumulative sum curve for the eigenvalues

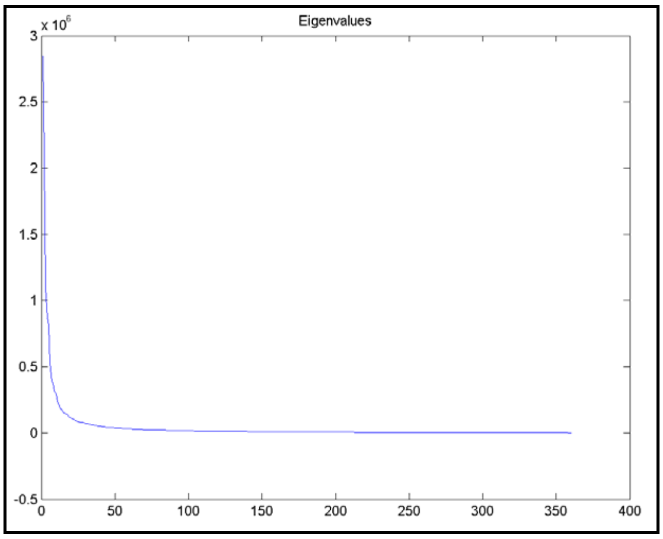

Figure 2.6. A typical eigenvalue spectrum

Eigenvectors can be considered as the vectors pointing in the direction of the maximum variance and the value of the variance that the eigenvector represents is directly proportional to the value of the eigenvalue (i.e. the larger the eigenvalue indicates the larger variance the eigenvector represents). Hence, the eigenvectors are sorted with respect to their corresponding eigenvalues. The eigenvector having the largest eigenvalue is marked as the first eigenvector, and so on. In this manner, the most generalizing eigenvector comes first in the eigenvector matrix.

In the next step, the training images are projected into the eigenface space and thus the weight of each eigenvector to represent the image in the eigenface space is calculated. This weight is simply the dot product of each image with each of the eigenvectors. To find the projection of a training image on each of the eigenvectors

$$\omega_k = \upsilon_k^T \cdot \Phi = \upsilon_k^T \cdot (\Gamma - \Psi) \tag{2.13}$$

where k = 1, 2, …, M’. Representing the training image in the eigenface space, we obtain the weight matrix as

$$\Omega = [\omega_1 \ \omega_2 \ \ldots \ \omega_{M'}]^T \tag{2.14}$$

At this point, the images are just composed of weights in the eigenface space, simply like they have pixel values in the image space. The important aspect of the eigenface transform lies in this property. Each image is represented by an image of size (N_x x N_y) in the image space, whereas the same image is represented by a vector of size (M’ x 1) in the eigenface space. Moreover, having the dimension structure related to the variance of the data in hand makes the eigenface representation a generalized representation of the data. This makes the algorithm a solution to the “curse of dimensionality” problem seen in the standard pattern recognition task [33].

When a new test image is to be classified, it is also mean subtracted and projected onto the eigenface space and the nearest mean algorithm is used for the classification of the test image vector in the standard eigenface method; that is, the test image is assumed to belong to the nearest class by calculating the Euclidean distance of the test image vector to the mean of each class of the training image vectors [28].

If possible, it is better to work on a database of more than one image per individual in order to increase the robustness to minor changes in expression, illumination and slight variations of view angles. A class of images for an individual can be formed and this class can be considered as the representative image vector of that class. For an individual having q_i images in the database, the average of the projections of each class is the mean of all the projected image vectors in that class. In mathematical terms, we can obtain the average class projections as

$$\Omega_{\Psi} = \frac{1}{q_i} \sum_{i=1}^{q_i} \Omega_i \tag{2.15}$$

This average class projection can be used as one of the vectors (representing an image class instead of an image vector) to compare with the test image vector. A similarity measure is defined as the distance between the test image vector and ith face class as

$$\delta_i = \left\| \Omega_T - \Omega_{\Psi_i} \right\| = \sqrt{\sum_{k=1}^{M_t} \left(\Omega_{T_k} - \Omega_{\Psi_{ik}} \right)^2} \tag{2.16}$$

A distance threshold may be defined for the maximum allowable distance from any face class, which is half of the distance between the two most distant classes as

$$\Theta = \frac{1}{2} \max \left(\left\| \Omega_{\Psi_i} - \Omega_{\Psi_j} \right\| \right) \tag{2.17}$$

The classification procedure of the eigenface method ensures that face image vectors should fall close to their reconstructions, whereas non-face image vectors should fall far away. Hence, a distance measure is defined as

$$\varepsilon^2 = \left\| \Phi - \Phi_f \right\|^2 \tag{2.18}$$

which is the distance between the mean subtracted image and the reconstructed image.

The recognition of an image knowing these two measures δ_i and ε can be done as follows:

- If $\varepsilon > \Theta$ $\rightarrow$ the image is not a face (independent of the values of δ_i),
- If $\varepsilon < \Theta$ and for all i $\rightarrow$ $\delta_i > \Theta$ $\rightarrow$ the image is an unknown face,

- If $\varepsilon < \Theta$ and for one of $i \rightarrow \delta_i < \Theta \rightarrow$ the image belongs to the training face class i.

This is the standard Eigenface approach suggested by Turk and Pentland in 1991 [28, 34].

Another developed algorithm related to PCA is the weighted kernel principal component analysis (WKPCA) which is also proposed for feature extraction [35]. Weights that represent inter-class relationships are incorporated into a kernel matrix. Images in the training and testing set are projected onto the subspace obtained from the weighted kernel matrix. The idea is to use genetic algorithms to select optimal weights. WKPCA based methods require low capacity and memory. They have high tolerance to noise and provide effective indexing on using small sized spaces. They can be affected from different environmental conditions since these methods are designed for good poses, constant luminance conditions and accurate and smooth captured images.

2.2.1.2. 3D Morphable Model

The main idea behind the morphable model approach is that given a sufficiently large database of 3D face models any arbitrary face can be generated by morphing the ones in the database [36, 37]. These methods are not sensitive to the external environmental parameters such as pose, orientation, and luminance. The 3D model of a face has all the pose specifications of the face. Therefore, the face gathered from the input image can be easily compared with the poses of the 3D model face, and can be recognized according to the one of the poses of this 3D model. These types of methods require the usage of computer technologies which provide 3D modeling. Advantage of the appearance-based methods is as follows. These methods are implemented successfully to images which have low quality and low resolution. The disadvantages of these methods are as follows. They require a number of different test images. Besides, illumination conditions and different pose angles make for the methods to model the face harder.

2.2.2. Feature-based Method

Feature-based methods cover not only shape or the relations between eyes and mouth; they cover all of the face data. These methods can be classified into three groups.

2.2.2.1. Feature-based Elastic Bunch Graph Matching (EBGM)

All human faces share a similar topological structure. All faces have critical points above them like nose tip, eyes, lips, ears, contour between brow and hairs etc. In these methods, faces are represented as graphs with nodes positioned at these points and edges labeled with 2D distance vectors. Figure 2.7 shows some sample graphs [38].

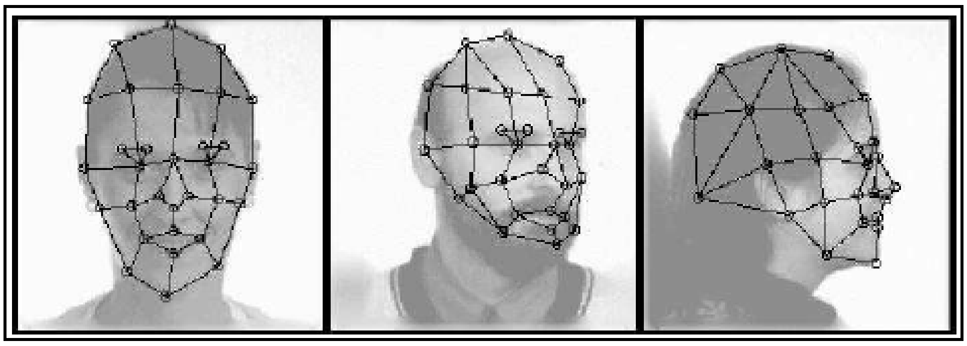

Figure 2.7. Elastic Bunch Graph Matching Method –Marked critical points and edges

First, critical points are obtained using Gabor filtering. Then, graphs are obtained between these critical points. After that, the graph patterns are compared and decisions are made for the recognition of the face pattern. Success rate of these methods is based upon the true localization of the important critical points of the human face. Therefore, many different algorithms are introduced to locate the critical points [34].

2.2.2.2. Hidden Markov Model (HMM)

HMM deals with sequences of coherent 1D signals (feature vectors), while an image is usually represented by a simple 2D matrix. However, this dilemma can be solved by applying a sliding window to the image covering the entire width of the image which is moved from the top to the bottom of the image. The intensity values of the pixels in the windows are passed as 1D feature vectors to the HMM process. Successive windows overlap to avoid cutting of significant facial features and to bring the missing context information into the sequence of feature vectors [39, 40]. The human face can be divided in horizontal regions like forehead, eyes, nose, mouth, chin, which are recognizable even when observed in isolation. Thus the face is modeled as a linear left-right HMM model of these five states.

The complete recognition system incorporates a preprocessing step followed by four processing steps. The preprocessing procedure includes the normalization of the image width, the histogram equalization, and the manual determination of the face location. It is applied to both database and test images. A test image that was not used for training first passes the preprocessing step. Then, the probability of producing this image is computed for every person, i.e. every model, in the database.

2.2.3. Combined Method

These methods are the combination of the feature-based and the appearance-based methods. They give higher success rates among the previous methods. Cootes *et. al.* [41] used the point distribution model and the local gray level which deals with the color changes in pixel in gray level. This model is applied to the PCA method. It creates a vector that stores the face data. The decision between the database and test images is done according to the obtained values using stored vector information.

Decrease in the complication which occurs by the environmental effects can be given as an advantage to feature-based methods. Disadvantages of these methods can be lined up as follows. To obtain face properties in a fully automatic way is not easy. Creation of models is a long, complicated and may be a cracked process since the local minimum values can give faulty results. Images in these methods should have high quality and high resolution.

2.3. TRACKING

The purpose of tracking an object is to generate the trajectory of a moving or non-moving body in time by locating its position in every video frame. Object tracker can also provide the whole region in the frame which is occupied by the object at every time instant.

The object tracking algorithm can run the tasks of detecting the object and establishing correspondence between the object instances across frames separately or jointly. It means that if the tasks work separately, first possible object regions in every frame are extracted with the object detection algorithm, then the algorithm will track these obtained object regions. In the latter case, the object region and correspondence is jointly estimated by iteratively updating object location and region information obtained from previous frames. In either tracking approach, objects are represented using appearance or shape models.

The model selected to represent object shape limits the type of motion. For example, in the case where a geometric shape representation like an ellipse or a circle is used for the object, parametric motion models like affine or projective transformations are appropriate. These representations are used for approximation of the motion of rigid bodies in the scene. However, for a non-rigid body, silhouette or contour representation is used since they include descriptive details of the body. If an object is represented as a point, then only a translational model can be used. Our face model in this study is also represented as a point. We use the centroid of the face region as a point descriptor.

Tracking will be considered in three main subsections as Kernel, Silhouette and Point Tracking. Figure 2.8(a) shows an example rectangle kernel tracking. Examples on Silhouette tracking can be seen in the Figures 2.8(b) and 2.8(c). An example of object correspondence as point tracking is shown in Figure 2.8(d).

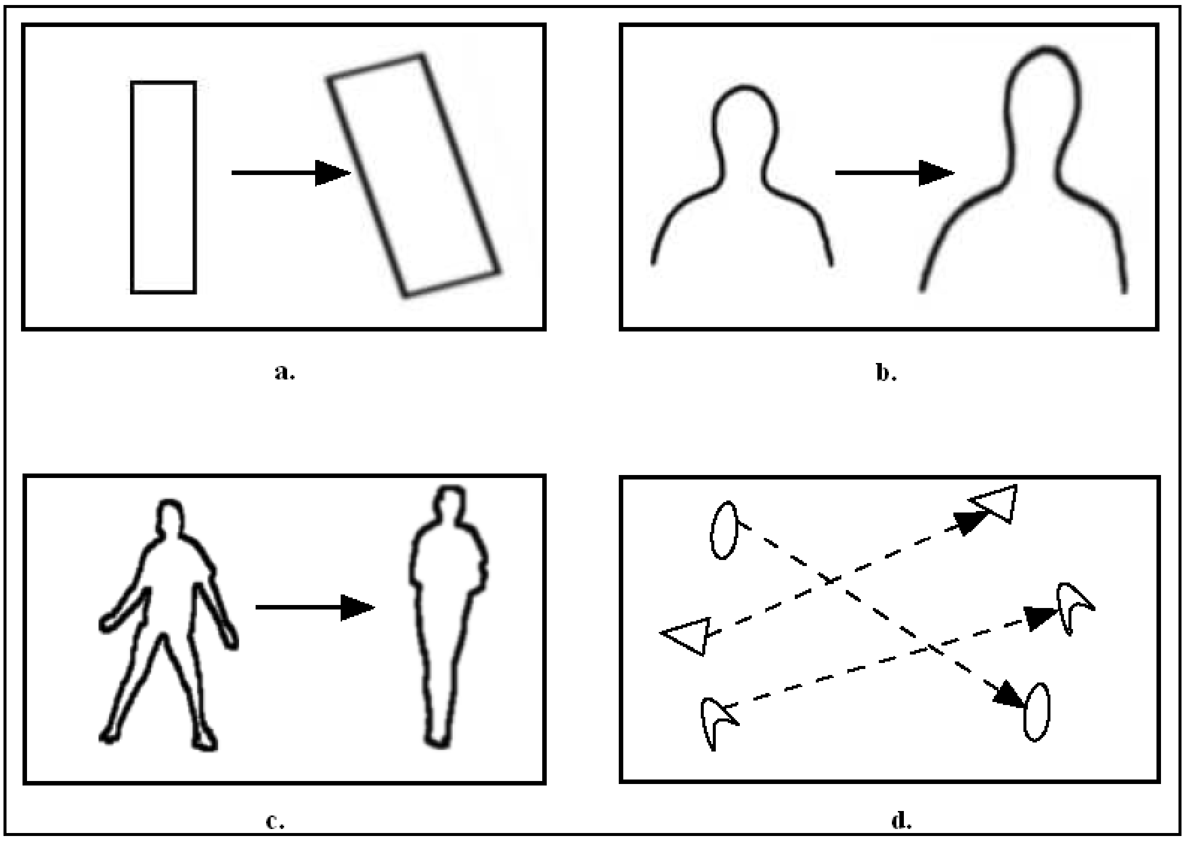

Figure 2.8. a. Rectangle kernel tracking, b. c. silhouette tracking, d. point tracking

2.3.1. Kernel Tracking

Kernel refers to the object appearance and shape. For example, the kernel can be elliptical shape or rectangular template with an associated histogram. Bodies in the consecutive frames are tracked by computing the motion of these types of kernel. This motion is usually in the form of a parametric transformation like translation, affine or rotation [42].

The main purpose of the trackers in this category is to estimate the object motion. Kernel tracking is typically performed by computing the motion of the object represented by a primitive object region in consecutive frames. These algorithms differ in terms of the appearance representation used, the number of objects tracked, and the method used to estimate the object motion. Kernel tracking is categorized in two groups as template and density-based appearance models and multi-view appearance models [42].

2.3.1.1. Template and density-based appearance models

The most common approach for track a single object is the template matching method. Template matching searches the image to obtain a region similar to the object template. The position of the template in the current image is computed by a similarity measure, for example, cross correlation. A disadvantage to template matching can be given as its high computation cost due to the brute force search. To reduce the computational cost, methods are usually limited to the object search to the vicinity of its previous position [43].

Other object representations can be used for tracking. For instance, color histograms or mixture models can be computed by using the appearance of pixels inside the rectangular or ellipsoidal regions. Such an object model is generated by finding the mean color of the pixels inside the rectangular object region [44].

Another method uses a weighted histogram computed from a circular region to represent the object [42]. In this method, instead of performing a brute force search for locating the object, they use the mean-shift procedure. The mean-shift tracker maximizes the appearance similarity iteratively by comparing the histograms of the object, and the window around the hypothesized object location. At each iteration, the mean-shift vector is computed such that the histogram similarity is increased. This process is repeated until convergence is achieved, which usually takes five to six iterations. An example of mean-shift tracking is given in Figure 2.9. An obvious advantage of the mean-shift tracker over the standard template matching is the elimination of a brute force search, and the computation of the translation of the object patch in small number of iterations. However, mean-shift tracking requires that a portion of the object is inside the circular region upon initialization (part of the object has to be inside the white ellipse in Figure 2.9(b)).

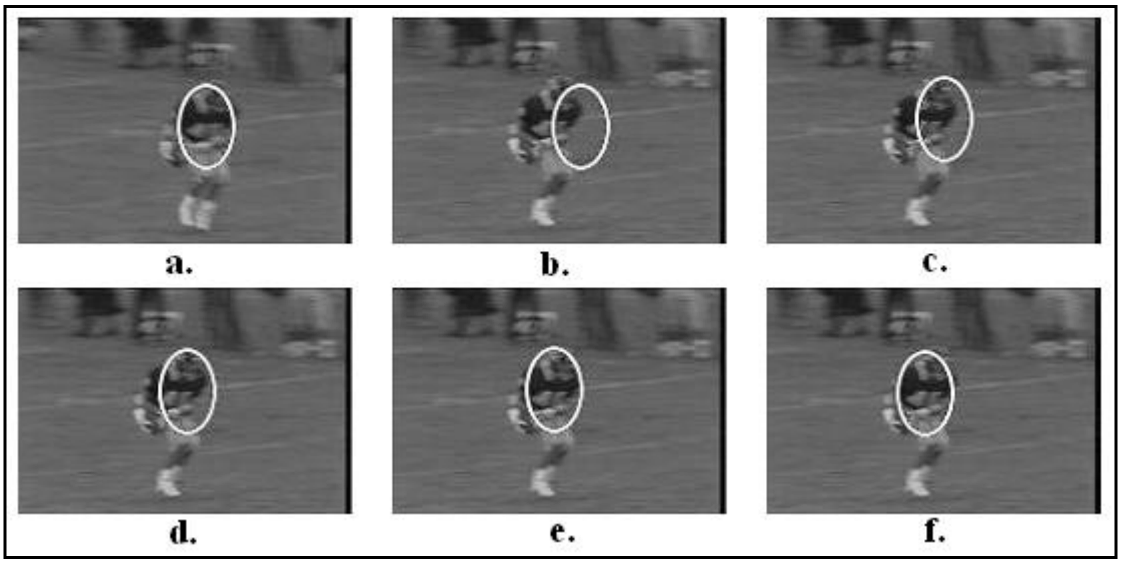

Figure 2.9. Mean-shift tracking iterations. a. estimated object location at time t − 1 b. frame at time t with initial location estimate using the previous object position c. d. e. location update using mean-shift iterations, (f) final object position at time t.

Another method tracks an object as a three component mixture, consisting of the stable appearance features, transient features and noise process [45]. The stable component identifies the most reliable appearance for motion estimation, that is, the regions of the object whose appearance does not quickly change over time. The transient component identifies the quickly changing pixels. The noise component handles the outliers in the object appearance that arises due to noise. Figure 2.10 shows the object tracking method used in this reference.

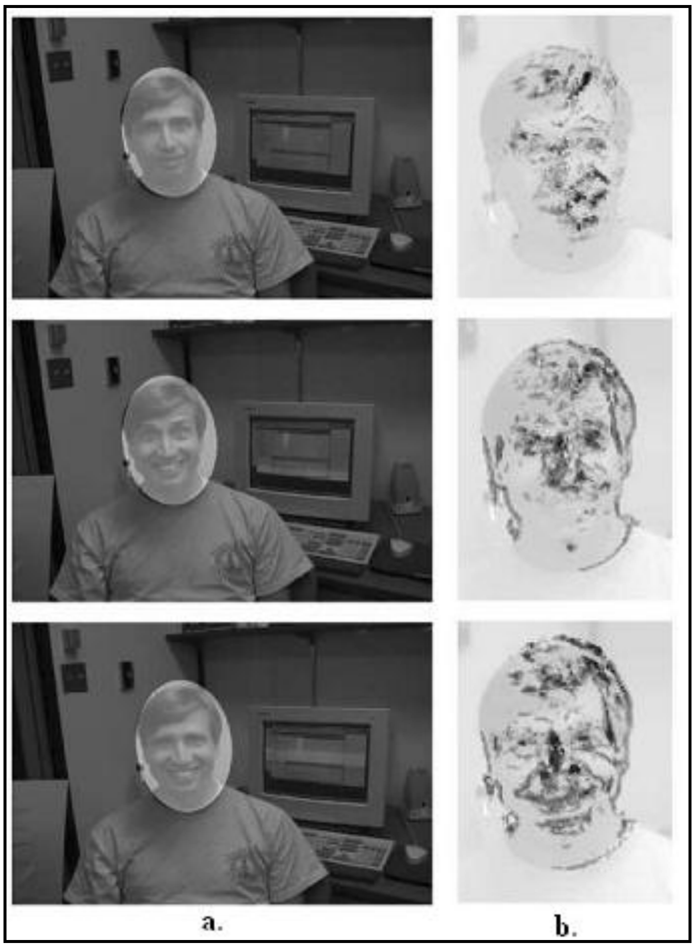

Figure 2.10. Results of the robust online tracking method [45] a. The target region in different frames. b. The mixing probability of the stable component. Note that the probabilities around the mouth and eyebrow regions change, while they remain the same in the other regions.

Another approach to track a region defined by a primitive shape is to compute its translation by use of an optical flow method. Optical flow methods are used for generating dense flow fields by computing the flow vector of each pixel under the brightness constancy constraint. Kanade-Lucas-Tomassi (KLT) feature tracker which iteratively computes the translation of a region centered on an interest point [46]. The results obtained by the KLT tracker are shown in Figure 2.11.

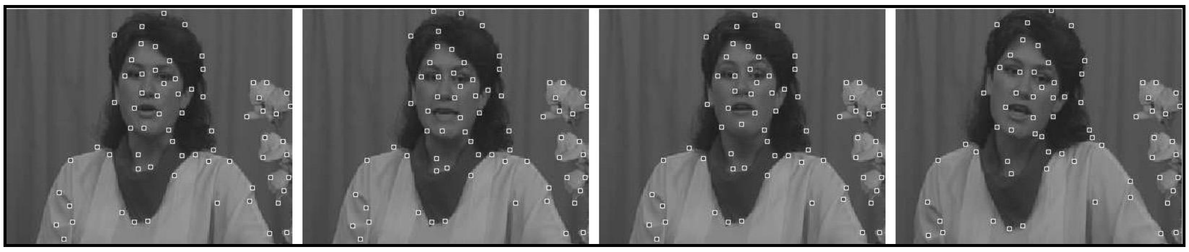

Figure 2.11. Tracking features using the KLT tracker.

Modeling objects individually does not take into account the interaction between multiple objects and between objects and background during the course of tracking. An example interaction between objects can be such as one object partially or completely occluding the other. The tracking methods given in the following paragraphs model the complete image, that is, the background and all moving objects are explicitly tracked.

Tao [47] proposed an object tracking method based on modeling the whole image as a set of layers. This representation includes a single background layer and one layer for each object. Each layer consists of shape priors (ellipse) motion model (translation and rotation), and layer appearance (intensity modeled using a single Gaussian). Layering is performed by first compensating the background motion modeled by projective motion such that the object's motion can be estimated from the compensated image using 2D parametric motion. Then, each pixel's probability of belonging to a layer (object) is computed based on the object's previous motion and shape characteristics. Any pixel far from a layer is assigned a uniform background probability. Later, the object's appearance (intensity, color) probability is coupled with the probability of each pixel to obtain the final layer estimate.

Another method, which consists of joint modeling of the background and foreground regions for tracking, is proposed by Isard and MacCormick [48]. The background appearance is represented by a mixture of Gaussians. Appearance of all foreground objects is also modeled by mixture of Gaussians. The shapes of objects are modeled as cylinders. Tracking is achieved by using particle filters.

2.3.1.2. Multiview appearance models

As it can be deduced from the name of the method, different views of the object are learned offline and used for tracking. The bodies in the scene may appear from different views. If the object view changes during tracking, the appearance model may no longer be valid, and the object tracking process might be lost. Therefore, multiview appearance models are used to overcome this problem.

Eigenspaces are used to compute the affine transformation from the current image of the object to the image reconstructed using eigenvectors [49]. First, a subspace representation of the appearance of an object is built using PCA, and then the transformation from the image to the eigenspace is computed by minimizing the so-called subspace constancy equation which evaluates the difference between the image reconstructed using the eigenvectors and the input image. Based on this, tracking is performed by estimating the affine parameters iteratively until the difference between the input image and the projected image is minimized.

In addition to this method, a Support Vector Machine (SVM) classifier is used for tracking [50]. During testing, the SVM gives a score to the test data indicating the degree of membership of the test data to the positive class. For SVM-based trackers, positive examples consist of the images of the object to be tracked, and negative examples consist of all that are not to be tracked.

2.3.2. Silhouette Tracking

Tracking is performed by estimating the object region in each consecutive frame. Silhouette tracking methods use the information encoded inside the object region. This information can be in the form of appearance density and shape models which are formed with the edge maps. Silhouettes are tracked by either shape matching or contour evolution [51].

Silhouette tracking is proposed when tracking of the complete body of an object is required [51]. The most important advantage of tracking silhouettes is their flexibility to handle a large variety of object shapes. Silhouettes can be represented in different ways.

The most common silhouette representation is in the form of a binary indicator function, which marks the object region by ones and the non-object regions by zeros. For contour-based methods, the silhouette is represented either explicitly or implicitly (Figure 2.12). Explicit representation defines the boundary of the silhouette by a set of control points. Implicit representation defines the silhouette by means of a function defined on a grid. The most common implicit contour representation is level sets [51]. Figure 2.12 gives an example for level sets based tracking.

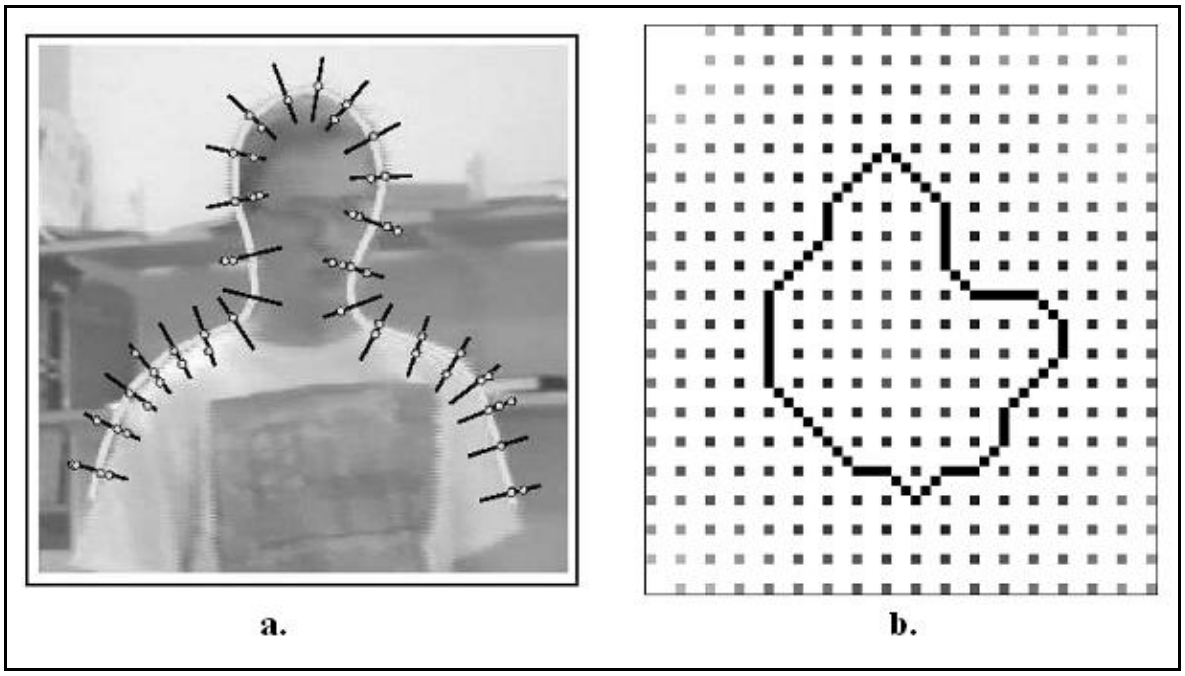

Figure 2.12. a. The measurements consist of image edges computed in the normal direction to the contour b. Level set contour representation, each grid position encodes the Euclidean distance between a grid point and the point on the contour; gray levels represent the values of the grid.

Occlusion handling is another important aspect of silhouette tracking methods. Usually methods do not address the occlusion problem explicitly. A common approach is to assume constant motion or constant acceleration where, during occlusion, the object silhouette from the previous frame is translated to its hypothetical new position.

Another important issue related to silhouette trackers is their capability for dealing with object split and merge. For instance, while tracking a silhouette of a person carrying an object, when the person leaves an object, a part of the person's contour will be placed on the left object (region split). These topology changes of region split or merge can be handled well by implicit contour representations [51].

2.3.2.1. Shape matching

Silhouette tracking can be executed by using shape matching. Shape matching is performed similar to tracking based on template matching where an object silhouette and its associated model are searched in the current frame. The search is performed by computing the similarity of the object with the model generated from the hypothesized object silhouette based on previous frame. In this approach, the silhouette is assumed to only translate from the current frame to the next. Therefore, non-rigid object motion is not explicitly handled. The object model, which is usually in the form of an edge map, is reinitialized to handle appearance changes in every frame after the object is located. This update is required to overcome tracking problems related to viewpoint and lighting condition changes as well as non-rigid object motion. Huttenlocher [52] performed shape matching using an edge-based representation. There are some other methods for shape matching which use Hough transform or color histograms [52, 53].

2.3.2.2. Contour tracking

Contour tracking methods, in contrast to shape matching methods, iteratively evolve an initial contour in the previous frame to its new position in the current frame. This contour evolution requires that some part of the object in the current frame overlap with the object region in the previous frame. Tracking by evolving a contour can be performed using two different approaches.

The first approach uses state space models to model the contour shape and motion [54 - 56]. The object's condition is represented in terms of the shape and the motion parameters of the contour. The state is updated at each time instant such that the contour's a posteriori probability is maximized. The posterior probability depends on the prior state and the current likelihood which is usually defined in terms of the distance of the contour from observed edges.

The second approach directly evolves the contour by minimizing the contour energy using direct minimization techniques such as gradient descent. The contour energy is defined in terms of temporal information in the form of either the temporal gradient (optical flow) [57, 58], or appearance statistics generated from the object and the background regions [59, 60]. In contrast to state space models, these methods can use implicit representations and allow topology changes. Figure 2.13 gives such an example.

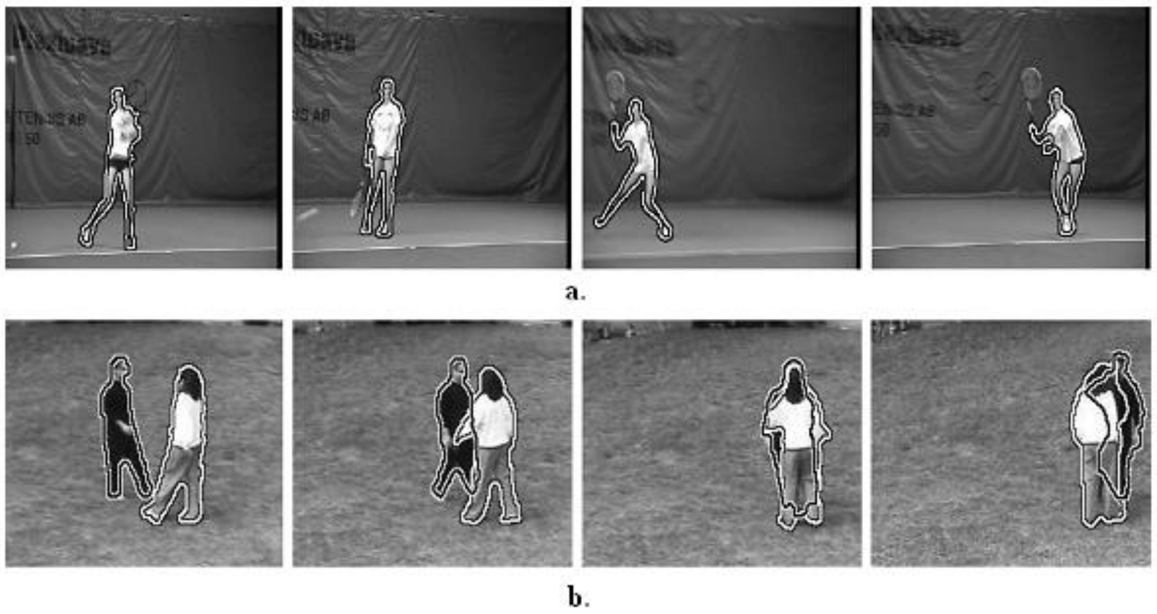

Figure 2.13. Contour tracking results a. tracking of a tennis player b. tracking in presence of occlusion [60]

2.3.3. Point Tracking

Bodies detected in consecutive frames are represented by points, and the association of the points is based on the previous body state which can include object position and motion. This approach requires an external mechanism to detect the bodies in every frame [61].

This type of tracking can be formulated as the correspondence of detected objects represented by points across frames. Point correspondence is a complicated problem, especially in the presence of entries and exits of bodies, occlusions, and misdetections.

Overall, point correspondence methods can be divided into two broad categories as deterministic and statistical. The deterministic methods use qualitative motion heuristics to constrain the correspondence problem. On the other hand, probabilistic methods explicitly take the object measurement and take uncertainties into account to establish correspondence.

2.3.3.1. Deterministic methods for correspondence

Deterministic methods for point correspondence define a cost of associating each object in the previous frame to a single object in the current frame using a set of motion constraints. Minimization of the correspondence cost is formulated as a combinatorial optimization problem. A solution, which consists of one-to-one correspondence (Figure 2.14(b)) among all possible associations (Figure 2.14(a)), can be obtained by optimal assignment methods, for example, Hungarian algorithm or greedy search methods [62]. The correspondence cost is usually defined by using a combination of the following constraints: proximity assumes the location of the object would not change from one frame to other (Figure 2.15(a)); maximum velocity defines an upper bound on the object velocity and limits the possible correspondence to the circular neighborhood around the object (Figure 2.15(b)); small velocity change (smooth motion) assumes the direction and the speed of the object which does not change very much (Figure 2.15(c)); common motion constrains the velocity of the object in a small neighborhood to be similar (Figure 2.15(d)). This constraint is usable for objects represented by multiple points; rigidity assumes that objects in the 3D world are rigid. Therefore, the distance between any two points on the actual object will remain unchanged (Figure 2.15(e)); proximal uniformity is a combination of the proximity and the small velocity change constraints.

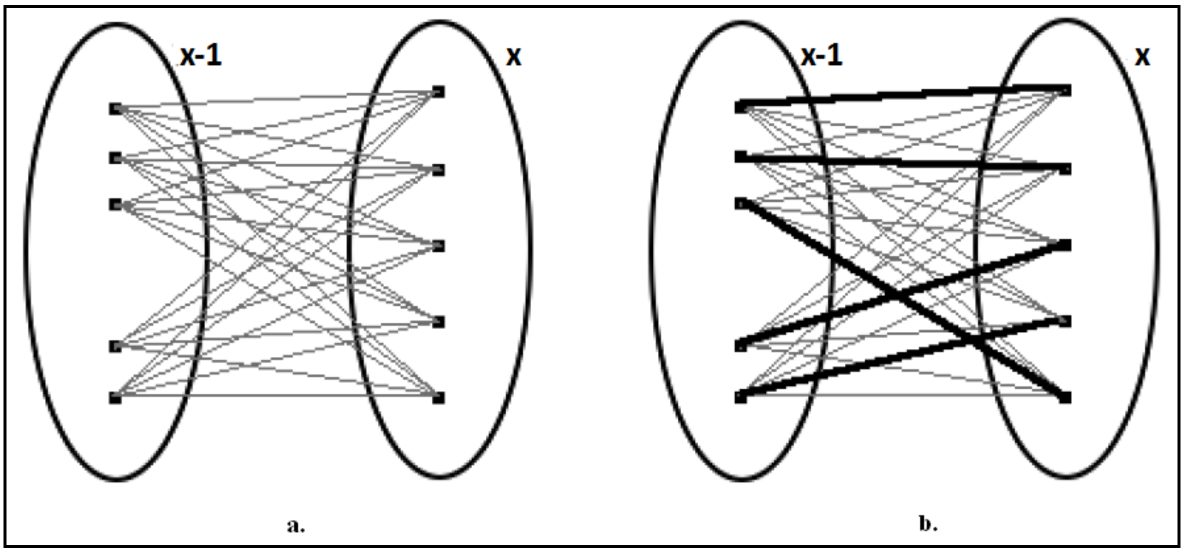

Figure 2.14. Point correspondence a. All possible associations of a point (object) in frame (x – 1) with points (objects) in frame x b. unique set of associations plotted with bold lines.

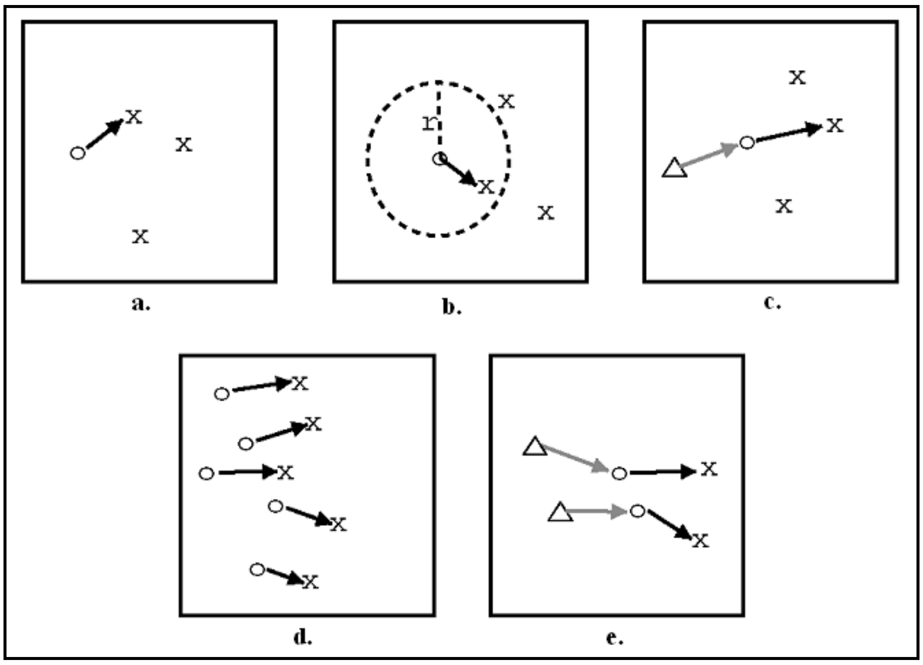

Figure 2.15. Different motion constraints a. proximity b. maximum velocity (r denotes radius) c. small velocity-change d. common motion e. rigidity constraints. Δ denotes object position at frame (x – 2), O denotes object position at frame (x – 1), and finally × denotes object position at frame x.

A method was proposed to solve the correspondence by a greedy approach based on the proximity and rigidity constraints [63]. A modified version of the same algorithm which computes correspondences in the backward direction (from the last frame to the first frame) in addition to the forward direction is also analyzed. This method cannot handle occlusions, entries, or exits. A new method was proposed to handle these problems by first establishing correspondence for the detected points and then extending the tracking of the missing objects by adding a number of hypothetical points [64].

A new approach proposes a method, which is constrained by proximal uniformity [65]. Initial correspondences are obtained by computing optical flow in the first two frames. The method does not address entry and exit of objects. If the number of detected points decreases, occlusion or misdetection is assumed. Occlusion is handled by establishing the correspondence for the detected objects in the current frame. For the remaining objects, position is predicted based on a constant velocity assumption.

Another method, which was proposed by extending the previous three works, introduces the common motion constraint for correspondence [66]. The common motion provides a strong constraint for coherent tracking of points that lie on the same object. However, it is not suitable for points lying on isolated objects moving in different directions.

A work by Intille [67], which uses a modified version of proximal uniformity approach above for matching object centroids, the objects are detected by using background subtraction. The authors explicitly handle the change in the number of objects by examining specific regions in the image, (for example, a door), to detect entries/exits before computing the correspondence.

2.3.3.2. Statistical methods for correspondence

The statistical correspondence methods use the state space approach to model the object properties such as position, velocity, and acceleration. Measurements usually consist of the object position in the image, which is obtained by a detection mechanism. Measurements also contain noise and there may be some perturbations over object motions like maneuvering vehicles. These methods solve these tracking problems by taking the measurements to model uncertainties into account during object state estimation. Object state estimation is classified into two groups as single or multiple object state estimation.

For the single object case, if noise in the measurements has a Gaussian distribution, then the optimal state estimate is given by the Kalman filter. If the object state is not assumed to be a Gaussian, then the state estimation can be performed by using particle filters [68, 7].

- A Kalman filter is used to estimate the state of a linear system where the state is assumed to be distributed by a Gaussian. Kalman filtering consists of two steps as prediction and correction. The Kalman filter has been often used in the tracking algorithm in the vision community. For example, Broida and Chellappa used the Kalman filter to track points in noisy videos [68].
- The particle filter is used to overcome the poor estimation problem of the Kalman filter, if the estimation does not follow Gaussian distribution [7].

The Kalman filter and particle filter described above assume a single measurement at each time instant, that is, the state of a single object is estimated. Tracking multiple objects requires a joint solution of data association and state estimation problems. The multiple object state estimation is dealing with solving the problem of the correspondence before the Kalman or particle filters can be applied. The simplest method to perform correspondence is to use the nearest neighbor approach. However, if the objects are close to each other, then there is always a chance that the correspondence is incorrect. An incorrectly associated measurement can cause the filter to fail to converge. There exist several statistical data association techniques to tackle this problem [69].

As we mentioned before, we used point tracking method in order to track the face location in the image. We take the centroid of the face region as a point indicator and track this indicator with basic point tracking method in order to be fast. With the location change of this indicator in consecutive frames, we can decide that which camera should be active or non-active. We will discuss this method in Chapter 3.

3. THE PROPOSED SYSTEM

In a surveillance system, there can be various setups in order to track a human in a certain area. People may think that it is enough to set a camera in a corner or somewhere in the area to visualize it. However, all the cameras have a limited range to capture the picture of the given area. Although one can think that one camera it is enough to capture the whole sight, for this situation one can only see the one side view of the scene. In this thesis, we have setup a human tracking system in order to capture a certain area with different side of view and the cameras are collaborating with each other.

Our system has two scenarios. The first scenario is the basic one to develop our system and investigate the right flow of the algorithms. This scenario has two cameras, which are located parallel with some distance between them, and both of them look at the same direction in order to see the whole sight view. Figure 3.1 shows the installation of the system.

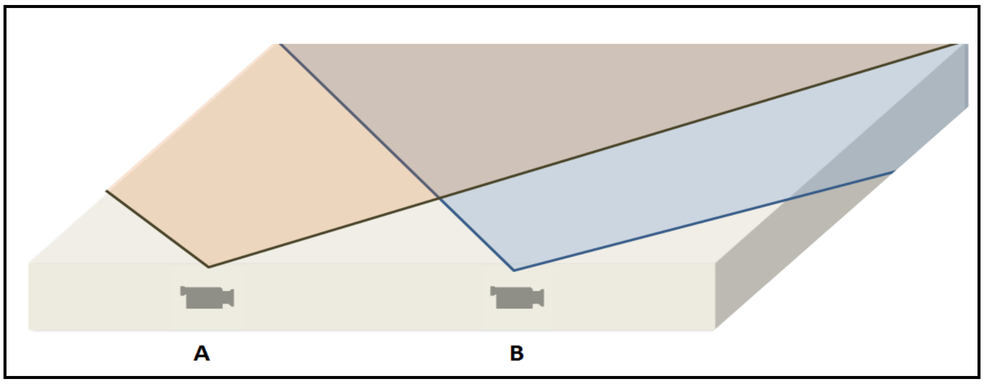

Figure 3.1. Camera settlement for the first scenario

The second scenario has four cameras; each of them is located at one of the four corners in a given area. For this scenario, the camera installation in the area is given in Figure 3.2.

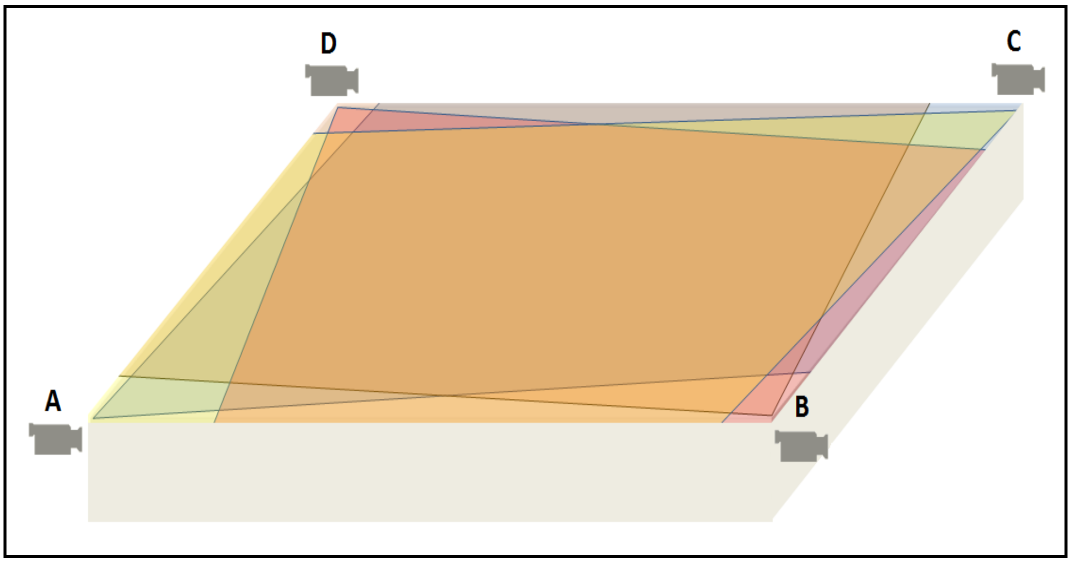

Figure 3.2. Camera settlement for the second scenario

Both scenarios use the same face detection, tracking and face recognition methods. In the following sections, we explain the face detection, tracking and face recognition methods we used in our system.

3.1. FACE DETECTION

As the face detection method, we choose the Adaboost algorithm in order to find the face in the image because its detection rate and speed is fast. However, Adaboost can sometimes find unexpected regions as face. In order to avoid this malfunction, we integrate skin color detection to Adaboost algorithm.

3.1.1. Adaboost Algorithm

The robust face detection systems are using boosting algorithms to detect objects both with high performance and in a very short time. The first boosting algorithm applied on face detection problem was AdaBoost (Adaptive Boosting) proposed by Viola and Jones [8]. The reason for choosing of this algorithm is its short time detection performance.

We started to investigate the Haar structures, proposed by Viola and Jones that using some filter-like objects to examine the image. The goal is to detect the alterations on gray color level for a given image and characterize the output for the pattern given on that image. The output of the image is done with a formula named *integral image*, which subtracts gray color level sums on contrast areas of the Haar filters. Figure 3.3 gives an example for integral images.

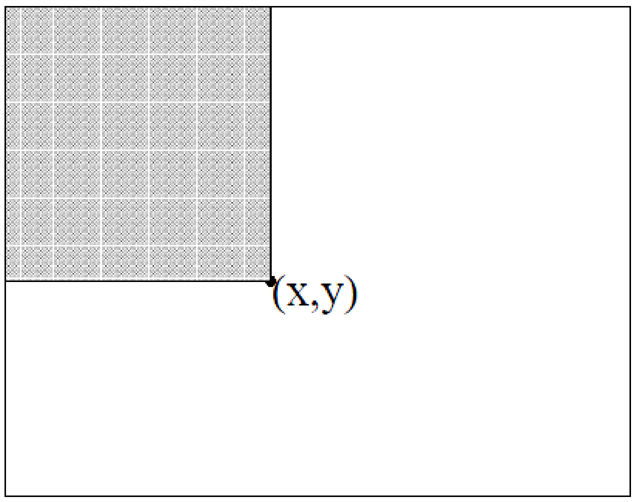

Figure 3.3. The value of the integral image at point (x,y) is the sum of all the pixels above and to the left.

The integral image at location (x,y) contains the sum of the pixels above and to the left of (x,y) as

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{3.1}$$

where ii(x,y) is the integral image and i(x,y) is the original image (see Figure 3.3). Using the following pair of recurrences:

$$s(x,y) = s(x,y-1) + i(x,y) \tag{3.2}$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \tag{3.3}$$

$$s(x,y) = s(x,y-1) + i(x,y) \tag{3.4}$$

where s(x,y) is the cumulative row sum, s(x,-1) = 0, and ii(-1,y) = 0, the integral image can be computed in one pass over the original image.

Using the integral image any rectangular sum can be computed in four array references (see Figure 3.4). Clearly the difference between two rectangular sums can be computed in eight references. Since the two-rectangle features defined above involve adjacent rectangular sums, they can be computed in six array references, eight in the case of the three-rectangle features, and nine for four-rectangle features.

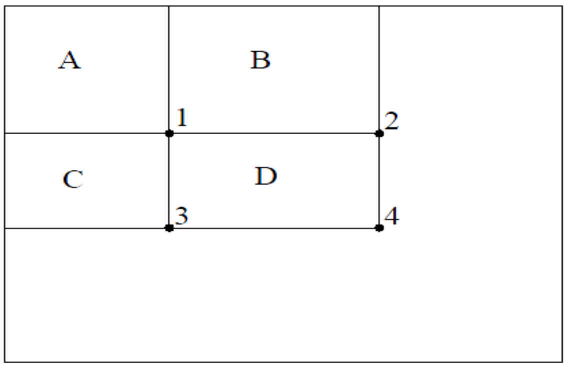

Figure 3.4. The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A+B, at location 3 is A+C, and at location 4 is A+B+C+D. The sum within D can be computed as 4+1-(2+3).

Using the integral image, the rectangular features can be calculated more efficiently. We try to first formulize Adaboost Algorithm and then explain more understandable how it works clearly below.

- Preparation and definition

Example images $(x_1,y_1),\ldots\ldots,(x_N,y_N)$ are given. N is the sum of a and b, where a is the number of positive examples and b is the number of negative examples. For positive examples $y_i = 1$ and for negative examples $y_i = 0$.

M is the number of weak classifiers. The number of maximum weak classifiers are represented with M_{max}.

```
➢ Start to algorithm
  • Initialize the weights:
      ○ IF (y_i = 1)
          ▪ THEN $w_i = \frac{1}{2a}$
      ○ IF (y_i = 0)
          ▪ THEN $w_i = \frac{1}{2b}$

➢ Loop for adding weak classifiers
  • FOR ( $M < M_{\max}$ )
      ○ Normalize weights: $w_i^{(M)} = \frac{w_i^{(M)}}{\sum_{j=1}^{N} w_j^{(M)}}$
      ○ IF (x_i is classified correctly)
          ▪ THEN e_i = 0
      ○ ELSE
          ▪ THEN e_i = 1 and $\beta_M = \frac{\varepsilon_M}{1-\varepsilon_M}$
      ○ $w_i^{(M+1)} = w_i^{(M)} (\beta_M)^{1-e_i}$
  • END FOR

➢ Output
  • IF $\sum_{m=1}^{M} \left[ w_m h_m(x) \right] \geq \frac{1}{2} \sum_{m=1}^{M} w_m$
      ○ THEN $h(x) = 1$
  • ELSE
      ○ THEN $h(x) = 0$
```

- RETURN h

Figure 3.5 shows the processed image after the Adaboost algorithm applied on it. A rectangle surrounds each of the face areas. Some misdetections tend to occur after the Adaboost processing. We will cover these errors with the application of the skin detection.

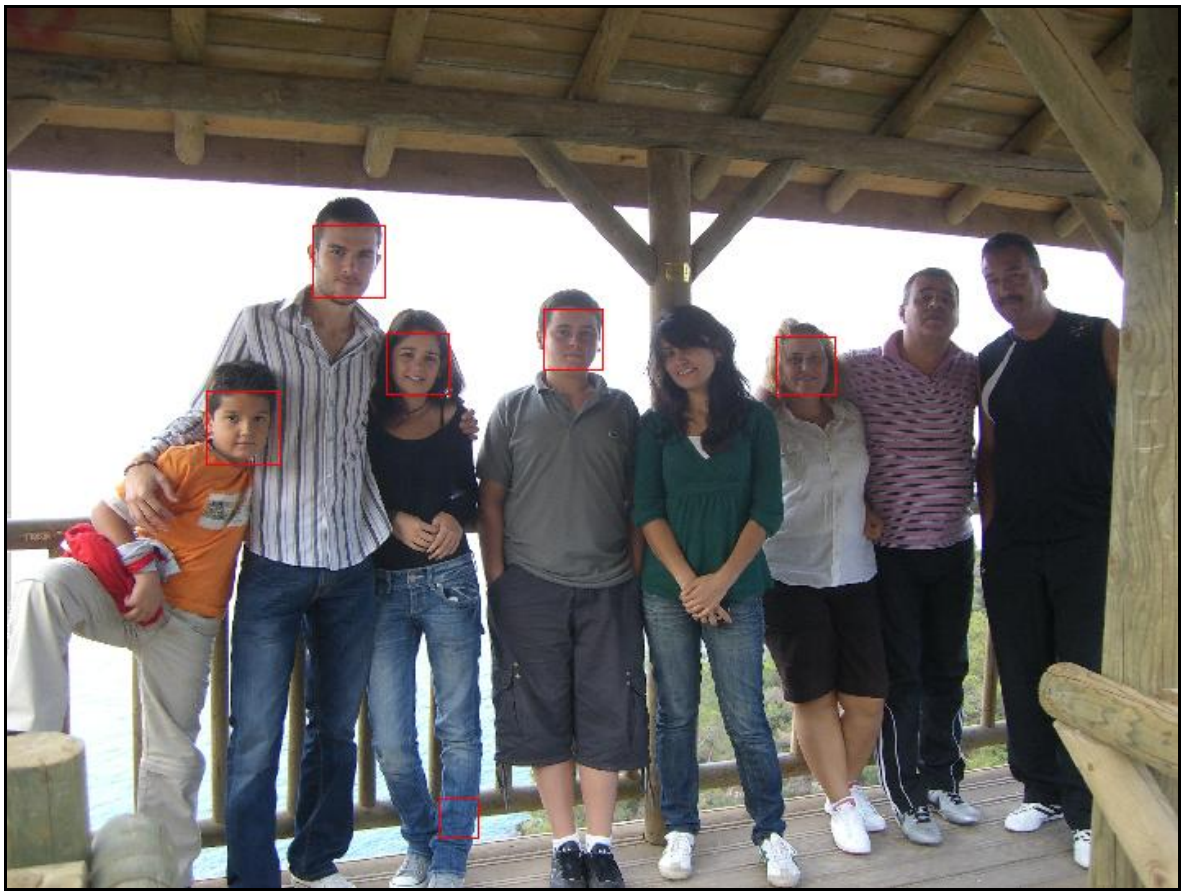

Figure 3.5. The processed image after the Adaboost algorithm

3.1.2. Skin Detection

In image processing, there are many color spaces in order to represent an image with these color space values. We have briefly cited them in Section 2.1.2.2. In this thesis, we used the YCbCr color space in order to extract the faces from the video image. YCbCr is one of two primary color spaces used to represent digital component video (the other is RGB). The difference between YCbCr and RGB is that YCbCr represents color as brightness and two color difference signals, while RGB represents color as red, green and blue. In YCbCr, the Y is the brightness (luma), Cb is blue minus luma (B-Y) and Cr is red minus luma (R-Y). We can see the CbCr color space plane in Figure 3.6. According to this plane, we can decide where a face skin color is located and how a threshold value can be estimated for a face skin color to be extracted from the image.

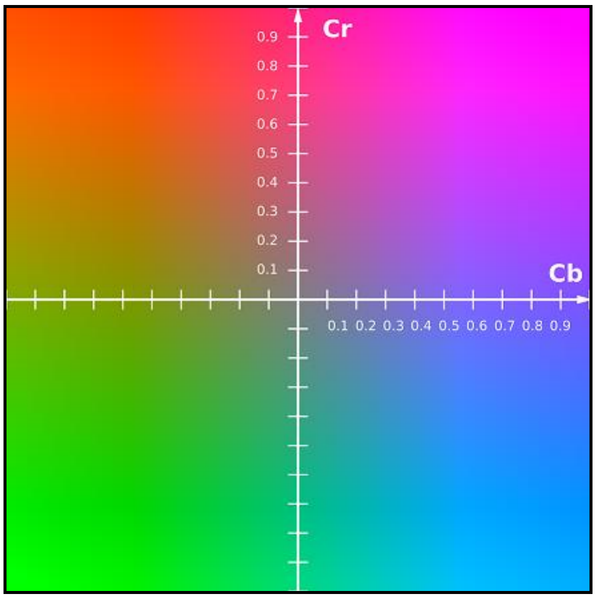

Figure 3.6. The CbCr plane at constant luminance Y=0.5

Below functions represent the basic conversion from an RGB image to YCbCr representation. Figure 3.7 shows the RGB image and its red, green, blue components and YCbCr image and its luma (Y), two chroma (Cr, Cb) components.

$$Y' = (0.299 \times R'_D) + (0.587 \times G'_D) + (0.114 \times B'_D) \tag{3.5}$$

$$C_B = 128 - (0.168736 \times R'_D) - (0.331264 \times G'_D) + (0.5 \times B'_D) \tag{3.6}$$

$$C_R = 128 + (0.5 \times R'_D) - (0.418688 \times G'_D) - (0.081312 \times B'_D) \tag{3.7}$$

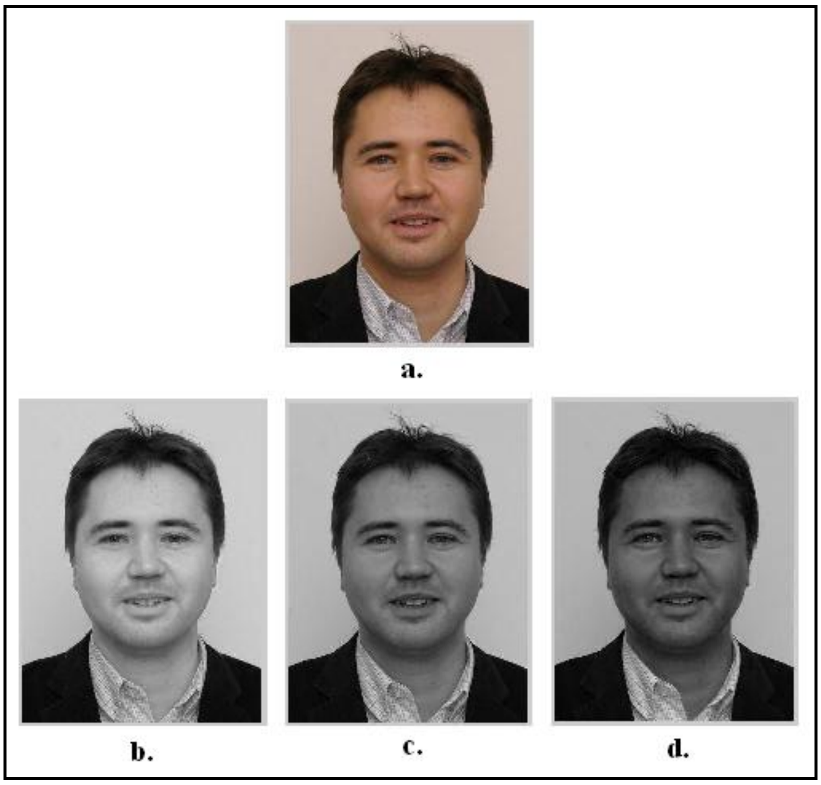

Figure 3.7. RGB color image and its components a. RGB image b. red component c. green component d. blue component

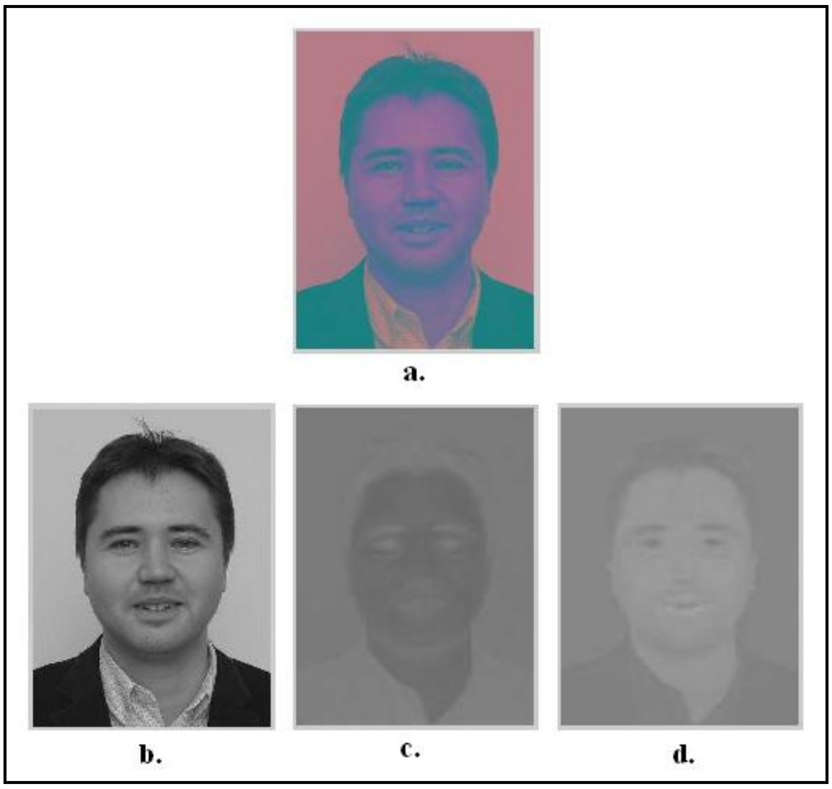

Figure 3.8. YCbCr color image and its components a. YCbCr image b. Y component c. Cb component d. Cr component

We have chosen the YCbCr color space since it provides the simplest model to define a region of skin tone pixels using Cr, Cb values. With carefully chosen thresholds, [Cr_1, Cr_2] and [Cb_1, Cb_2], a pixel is classified to have skin tone if its values (Cr, Cb) fall within the ranges $Cr_1 \leq Cr \leq Cr_2$ and $Cb_1 \leq Cb \leq Cb_2$. As can be seen in Figure 3.8 part D, the skin region in image is shining, and its histogram is shown in Figure 3.9. Threshold value could be easily found from the histogram figure. With successful thresholding the Cr component image, we can almost reach the face region as in Figure 3.10.

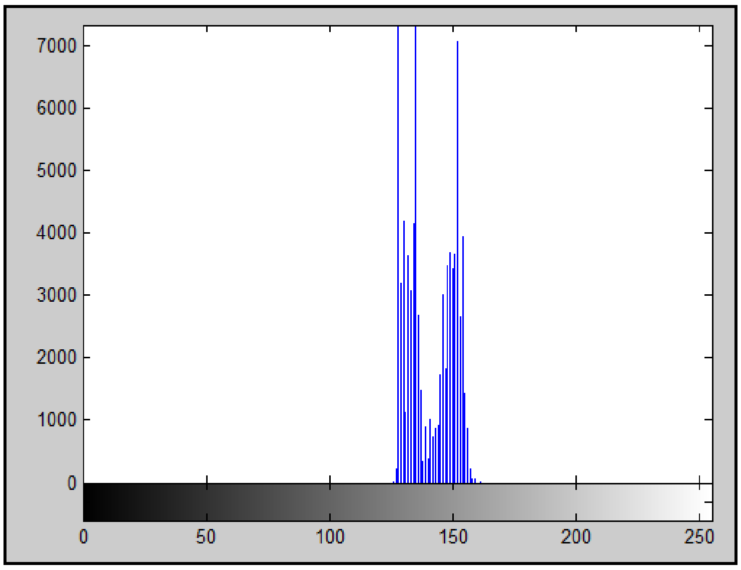

Figure 3.9. Histogram of skin color in Cr component of the YCbCr image

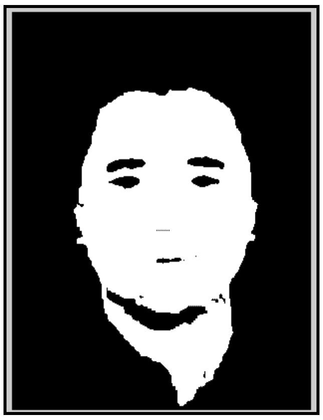

Figure 3.10. Face area after thresholding

After some morphological operations (dilation, erosion), face area can be marked out. These operations have been performed in order to remove the noisy pixels in face image. Figure 3.11 shows the last state of the image.

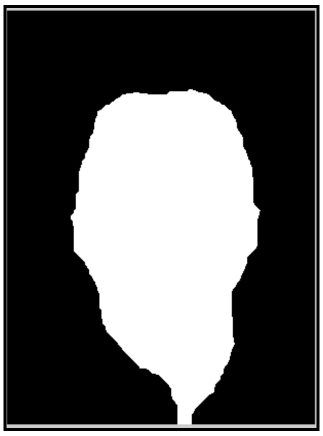

Figure 3.11. Final state of face image

Figure 3.12 shows the skin detection result, when a skin detection algorithm is applied to the image after the Adaboost algorithm applied to it. Both the Adaboost and skin detection methods are combined to improve the accuracy of locating faces in an image. We will cover this combination in Section 3.4.

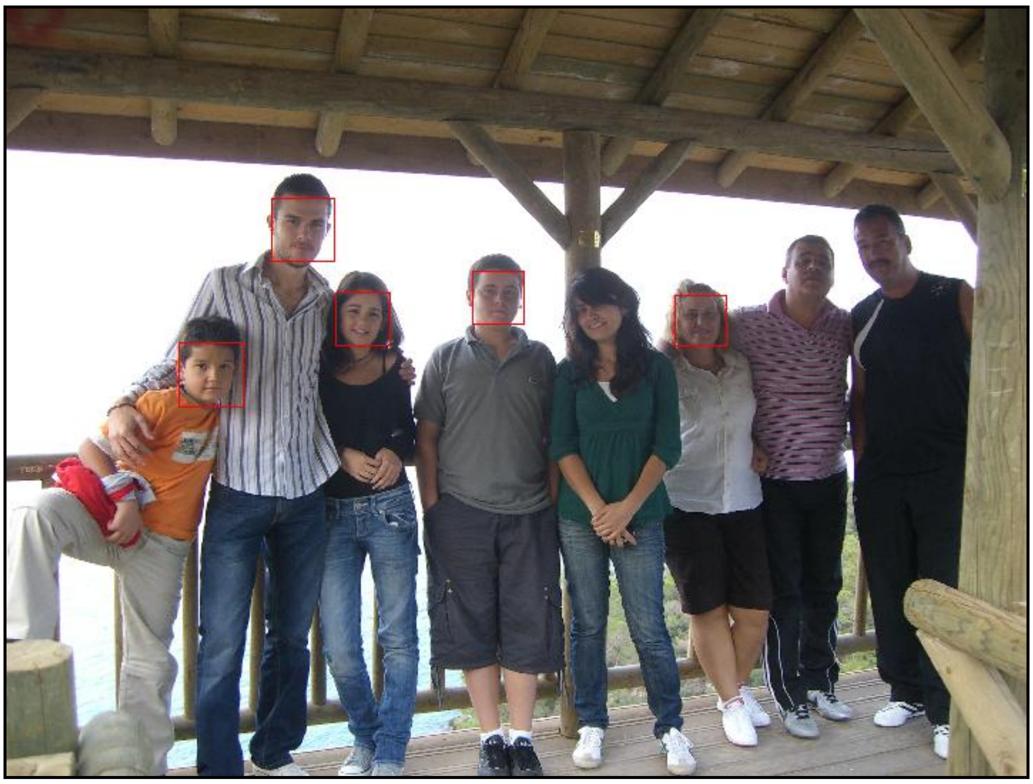

Figure 3.12. The skin detection result

3.2. TRACKING

We used the point tracking method in order to track the face in consecutive frames. Tracking point is chosen as the centroid of the extracted face. Moving directions of the face can be easily tracked by following the movement of this point. Before the centroid of the face is determined, a labeling operation has to be performed to the extracted face area. Connected binary pixels form a labeled area. Each face area is composed of these binary pixels.

In this thesis, we define the centroid of the face area as the midpoint of the cropped face image. Adaboost algorithm finalizes the processed image with cropped face images. The point indicator is followed horizontally if the face moves left or right. In both the two scenarios in our thesis, we use the horizontally tracking of the face. The flow diagram which was drawn with Smartdraw [70] in Figure 3.13 is constructed for the first scenario with two cameras.

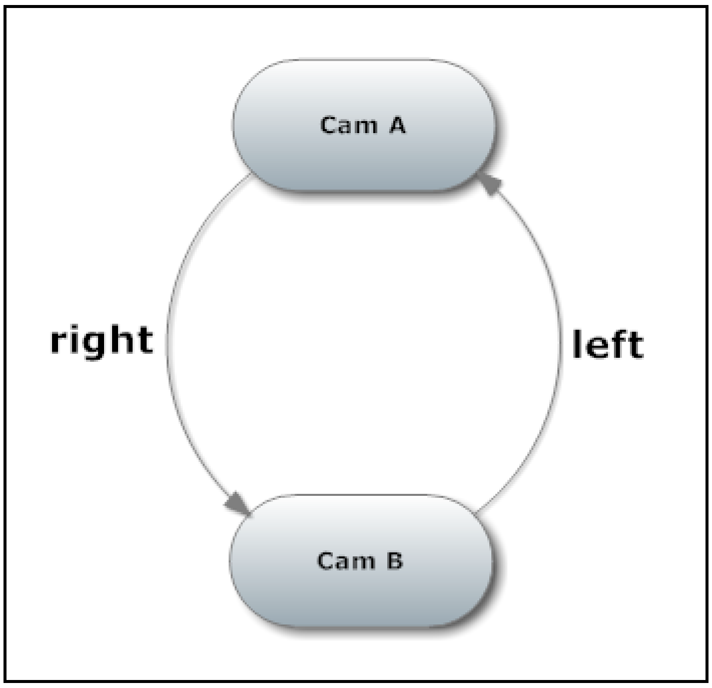

Figure 3.13. Flow diagram of the first scenario [70]

In this scenario, during the capture turn of camera A, if the point indicator moves to right, the camera B will be active and start to capture the frame and looks for the face and track it. In this step, the camera A will be inactive. After that if the tracked face moves to left, the camera A will be active and start to capture and camera B will fall to inactive mode.

The second scenario is more difficult to implement. Figure 3.14 shows the flow diagram of second scenario of the tracking face in four cameras system which was drawn with Smartdraw [70].

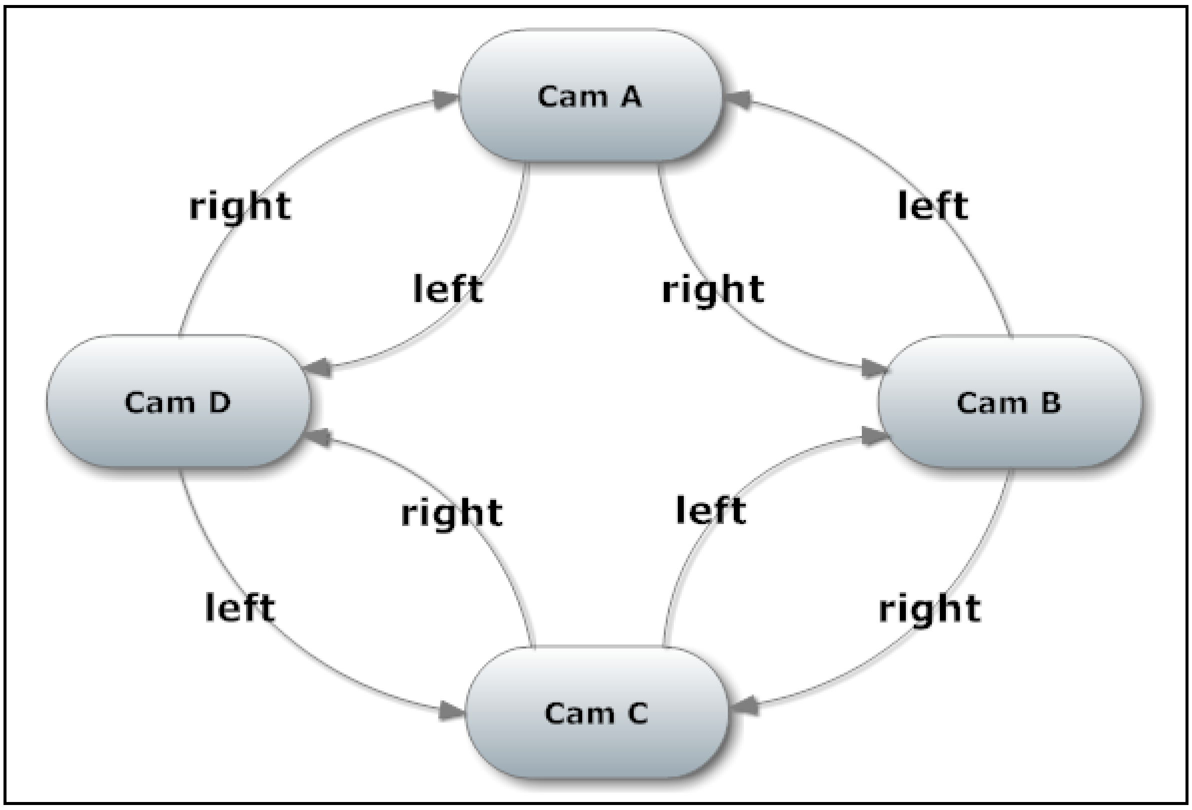

Figure 3.14. Flow diagram of second scenario [70]

The working principle of this scenario is like in first scenario, but it is improved as if a human turns back rapidly and active camera cannot detect the face, algorithm starts to find the face in each of four cameras. When the face is found by any cameras, the algorithm continues to detect the face from this camera’s captured frame.

3.3. FACE RECOGNITION

If a face is detected and tracked, it continues to be recognized with an algorithm integrated to the system. We have chosen the PCA algorithm to recognize the face. PCA method was introduced in Chapter 2. This algorithm looks for that if the tracking face is pair with the face in database. In our system, we have built a database with my friends and my family members. However, we also try the PCA algorithm success with generic databases introduced in image processing world. Figure 3.15 shows the database images gathered from Face Recognition Homepage [71].

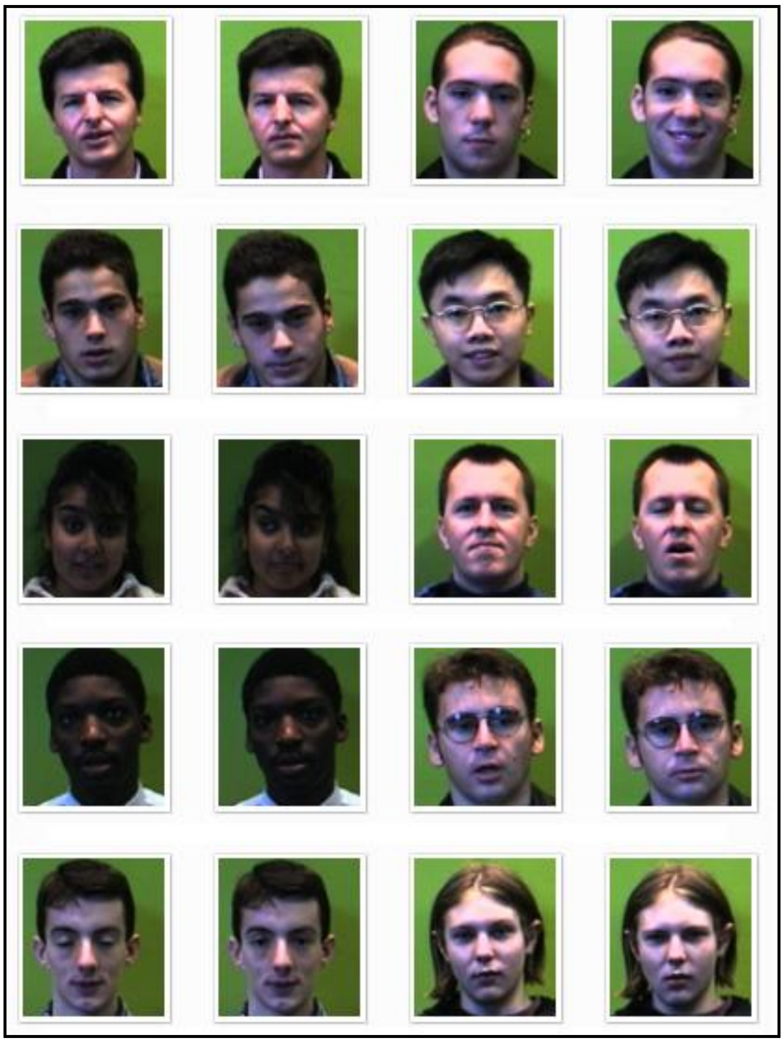

Figure 3.15. Database images gathered from Face Recognition Homepage [71]

Purpose of eigenface method is based on obtaining an eigenface space by applying this method to the database training images. Moreover, after the projection of the test image onto this eigenface space, distance of the projected test image to the database training images can be used to identify the test image.

All the images in the database and the tracked and cropped face are resized to 180 pixels in x coordinate and 200 pixels in y coordinate. This resizing is performed in order to equal the size of tracked images and its corresponding face in database.

The images in database are in RGB format and are in 200x180 size format. First, the database images are converted to grayscale. Then, these 2D images are converted into 1D image vectors in size of 36000x1. Then, we bring up each training face image vectors (n pieces) together in one matrix (T) in size of 36000xn. The mean value of T matrix is calculated and formed a matrix of m in size of 36000x1. Then, the deviation of each database image from mean image is calculated and a matrix A is obtained in size of 36000xn.

After this step, covariance matrix should be calculated, which have a formula of AxA’. However, its size will be very huge to work on it. Therefore, instead of working on a matrix in size 36000x36000, we can take the surrogate of this covariance matrix as A’xA and get a matrix L in size nxn. If we perfom the eigen process on matrix L, we will get two matrices of D and a eigenvector matrix. The diagonal elements of D are the eigenvalues for both AxA’ and A’xA.

Eigenvectors of covariance matrix (or so-called Eigenfaces) can be recovered from the eigenvectors of L. Multiplying the deviation matrix A with the eigenvector matrix of L will result the Eigenfaces matrix.

All centered images are projected into facespace by multiplying in Eigenface basis’s. Projected vector of each face will be its corresponding feature vector. Multiplying the transpose of Eigenfaces matrix with matrix A gives the projected image matrix.

After this point, if a test image should be classified, it has to be also mean subtracted and projected onto the eigenface space. The nearest mean algorithm can classify the test image vector in the standart eigenface method.

We consider the PCA algorithm in two pieces as initialization step and running step during the execution of the system.

The flow of initialization of our PCA algorithm is introduced below in algorithmic form.

```
➢ Form a matrix T using database images
  • FOR (i = 1 : n)
      o Convert ith RGB image in database to grayscale image
      o Convert 2D grayscale image matrix (200x180) to 1D image vector (36000x1)
      o T (i,:) = ith 1D image vector
  • END FOR
➢ Calculate the mean value of matrix T
  • FOR (i = 1 : 36000)
      o $m(1,i) = \frac{x(1,i) + x(2,i) + x(3,i) + ... + x(n,i)}{n}$
  • END FOR
➢ Calculate deviation
  • FOR (i = 1 : n)
      o $A(i,:) = T(i,:) - m$
  • END FOR
➢ Calculate L = A'xA
➢ Find eigenvalues of L
➢ Eliminate some eigenvalues, which are below the threshold
➢ Form new L, after elimination
➢ Calculate Eigenface matrix F = AxL
➢ Form projected centered image matrix P
```

```
• FOR (i = 1 : n)
    ○ $P(i,:) = F'xA(i,:)$
• END FOR
```

This initialization is performed for a one time before the system begins to run. After a face is detected and tracked, if the face will be recognized, the rest part of the PCA algorithm is executed.

The running step of the algorithm is explained below in algorithmic form.

```
➢ Get Red component of cropped (test) image
➢ Convert 2D Red image matrix (200x180) to 1D image vector (36000x1)
➢ Calculate deviation
  • $D = ImageVector - m$
➢ Calculate Test image feature vector
  • $V = F'xD$
➢ Drive an Euclidean distance matrix E
  • FOR (i = 1 : n)
      ○ $E(i) = (normalize(V - P(:,i)))^2$
  • END FOR
➢ Decide which column of E has minimum value (a = $3*10^{20}$)
  • FOR (i = 1 : n)
      ○ IF $\mathrm{E}(\mathrm{i}) < \mathrm{a}$
      ○ THEN $\mathrm{a=E}(\mathrm{i})$ and $j = i$
  • END FOR
  • Return j
➢ Decide if the test image corresponds to a face in database
  • IF $\mathrm{E}(j) < 3*10^{15}$
      ○ Return jth database image
  • ELSE
      ○ Put test image into null-class
```

3.4. THE OVERALL SYSTEM

In this chapter, we cover the overall system including the initialization of the system, execution of the system and results. We used Matlab for implementation of our system. Figure 3.16 shows the flow diagram of our system which was drawn with Smartdraw [70].

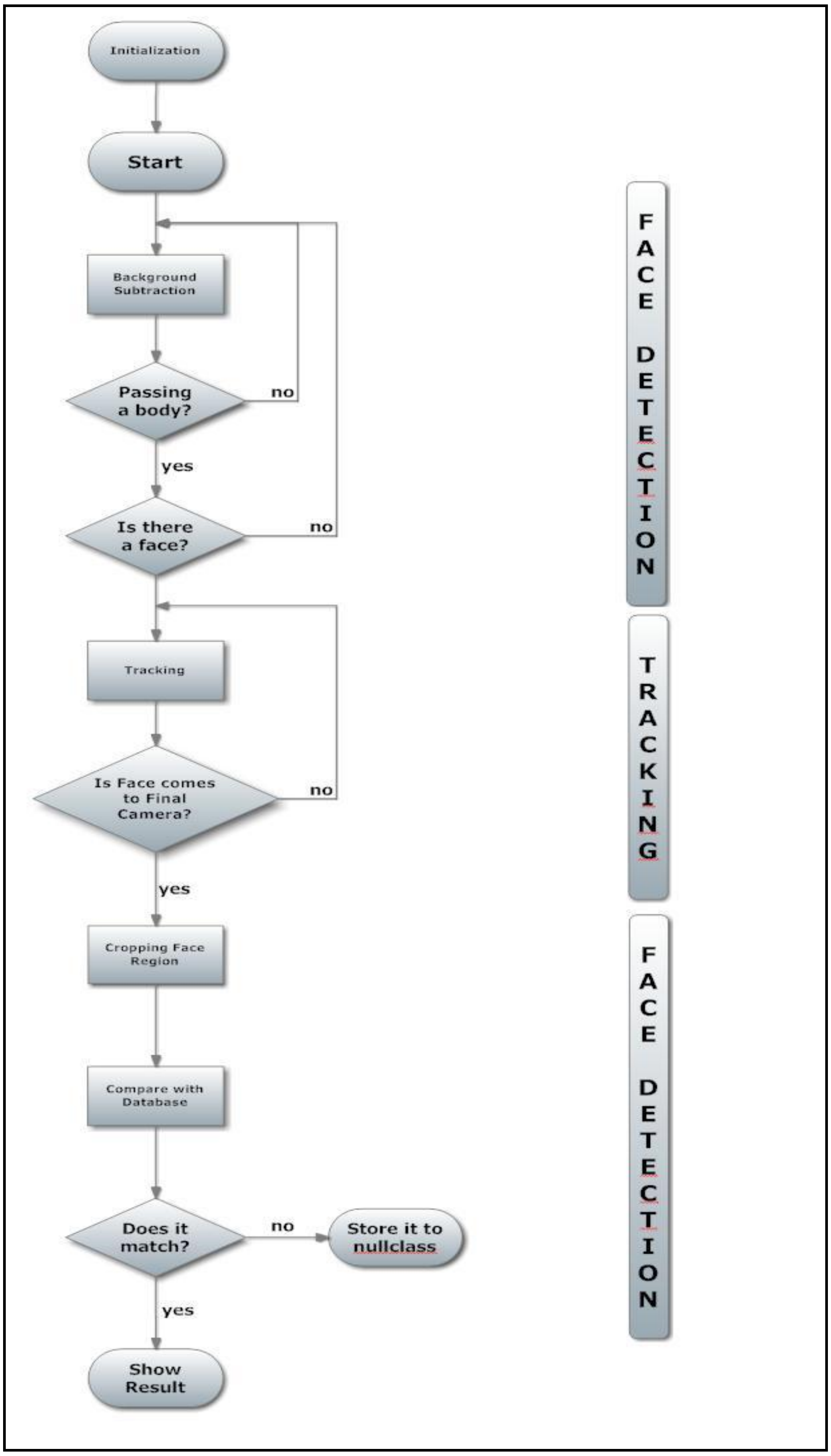

Figure 3.16. Flow diagram of the system [70]

3.4.1. Initialization of The System

Initialization step includes the database training for PCA algorithm and getting the median of background for robust background image. We have covered the database training before in chapter 3.3. We will explain a little bit why we get the median of the background image.

In real world, there will be always illumination changes in the environment. Therefore, we capture a small video of the free indoor space consists of totally 50 frames. Free indoor space means that there is no movement during the capturing of the video. Then, we process this video to take the median of the pixels of all 50 frames. The process of taking median of a video is explained below in algorithmic form.

```
➢ Store the background video in a 4D matrix
➢ Convert each RGB image in 4D matrix to grayscale image
➢ Take median of corresponding pixels in each image
  • Constitute a background image matrix with this median process
```

3.4.2. Execution of The System

```
➢ Background subtraction
  • IF (there is a movement)
      o GO TO b
  • ELSE
      o GO TO a
➢ Adaboost Algorithm
  • IF (there is a face)
      o GO TO c
  • ELSE
      o GO TO a
➢ Skin detection
  • Threshold the Cr component of images found in Adaboost
```

- IF (image contains a face)
    - GO TO Tracking
- ELSE
    - GO TO a

After we continue to track the face, we have to consider the tracking algorithm in two distinct parts. As we mentioned before, we have a system with two cameras and a system with four cameras.

First implementation is built with two cameras of two webcams. Second implementation is built with four cameras. Two of four cameras are webcams and the other two cameras are Sony CCD Color Cameras.

During the tracking of the face, we also want to recognize the face. We will also explain the working principle separately for our two systems.

3.4.2.1. Tracking with two cameras

Our system starts to detect the movement in the area and the face with the first camera, we call it as Camera A. The second one is called Camera B. The settlement was shown in Chapter 3. Figure 3.17 shows how two cameras display the area.

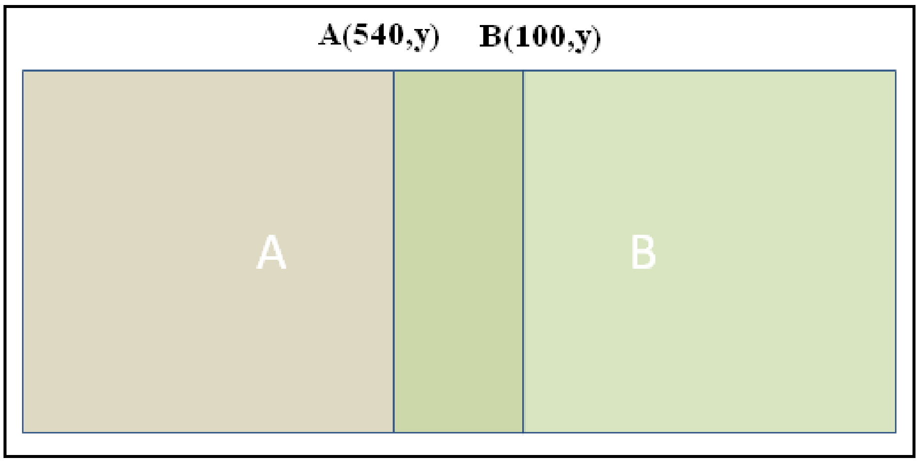

Figure 3.17. Displays of two cameras A and B

If the direction of the face is towards to Camera A and the face is tracked by the Camera A, our tracking algorithm starts to follow the movement of the face in x dimension. Figure 3.18 shows the flow diagram of the tracking algorithm for two cameras which was drawn with Smartdraw [70].

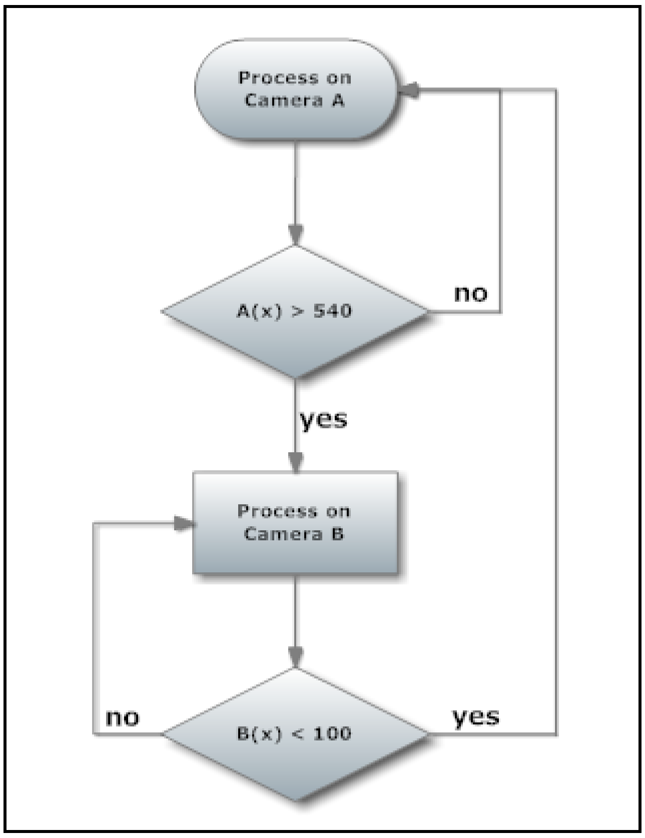

Figure 3.18. Flow diagram of tracking algorithm for two cameras [70]

Recognition is performed during tracking, if the human body comes to door, which is near the camera B. Decision on recognition will be introduced in the algorithm below.

```
➢ Track the face with Camera A
  • IF $A(x) > 540$
        ○ GO TO b
  • ELSE
        ○ IF (there is no face)
              ▪ GO TO b
```

```
          o ELSE
              ▪ GO TO a
➢ Track the face with camera B
  • IF $B(x) > 100$
          o IF (there is no face)
              ▪ GO TO a
          o ELSE
              ▪ IF $B(y) > 150$
                  • THEN Face Recognition
              ▪ ELSE
                  • GO TO b
  • ELSE
          o GO TO a
```

If the body turns back, and no face can be found during tracking, we start to capture the frame with other camera. It goes vice versa, until the face is found again.

3.4.2.2. Tracking with four cameras

Our system starts to detect the movement in the area and the face with the first camera, like the system with two cameras, called Camera A. The other ones are called Camera B, C and D respectively. The settlement for this system was also shown in the beginning of Chapter 3. Figure 3.19 shows how four cameras can cover and displays the whole area.

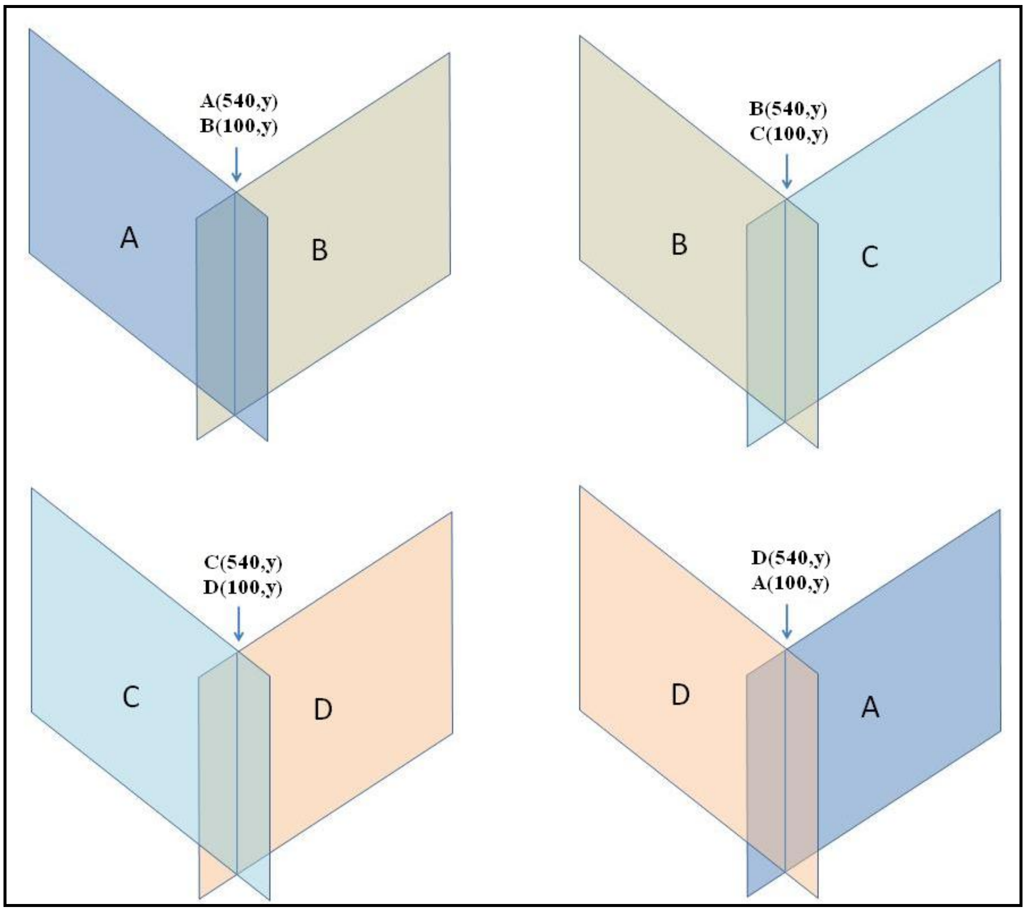

Figure 3.19. Displays of four cameras A, B, C and D

Recognition is performed during tracking, if the human body comes to door, which is near the camera D. Decision on recognition will be introduced in the algorithm below and also shown in Figure 3.20.

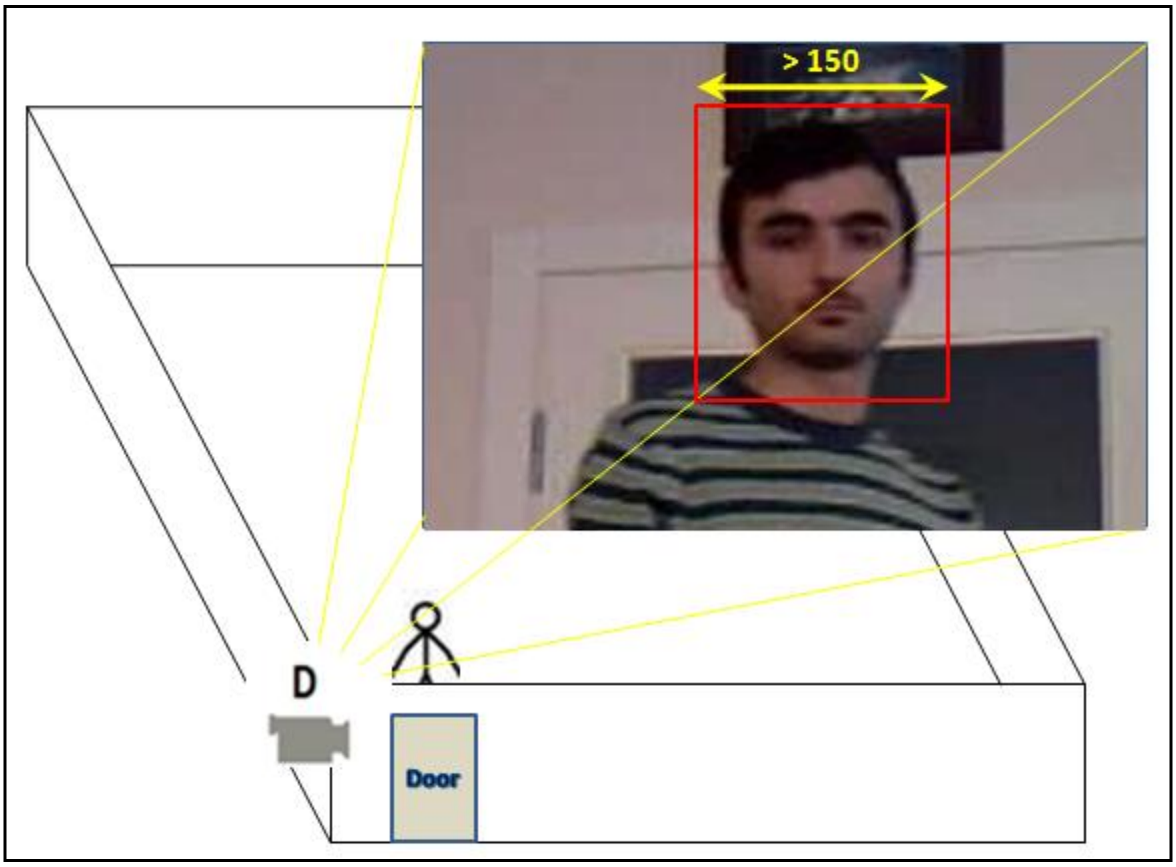

Figure 3.20. The decision for the recognition in four camera system

```
➢ Track the face with Camera A
  • IF $A(x) > 540$
      o GO TO b
  • ELSIF $100 < A(x) < 540$
      o IF (there is no face)
          ▪ GO TO c
      o ELSE
          ▪ GO TO a
  • ELSE
      o GO TO d
➢ Track the face with camera B
  • IF $B(x) > 540$
      o GO TO c
  • ELSIF $100 < B(x) < 540$
      o IF (there is no face)
          ▪ GO TO d
      o ELSE
          ▪ GO TO b
  • ELSE
      o GO TO a
➢ Track the face with Camera C
  • IF $C(x) > 540$
      o GO TO d
  • ELSIF $100 < C(x) < 540$
      o IF (there is no face)
          ▪ GO TO a
      o ELSE
          ▪ GO TO c
  • ELSE
      o GO TO b
➢ Track the face with camera D
  • IF $D(x) > 540$
      o GO TO a
  • ELSIF $100 < D(x) < 540$
      o IF (there is no face)
          ▪ GO TO b
      o ELSE
          ▪ IF $D(y) > 150$
              • THEN Face Recognition
          ▪ ELSE
              • GO TO d
  • ELSE
      o GO TO c
```

Figure 3.21 shows the flow diagram of working principle of the tracking algorithm for four cameras which was drawn with Smartdraw [70].

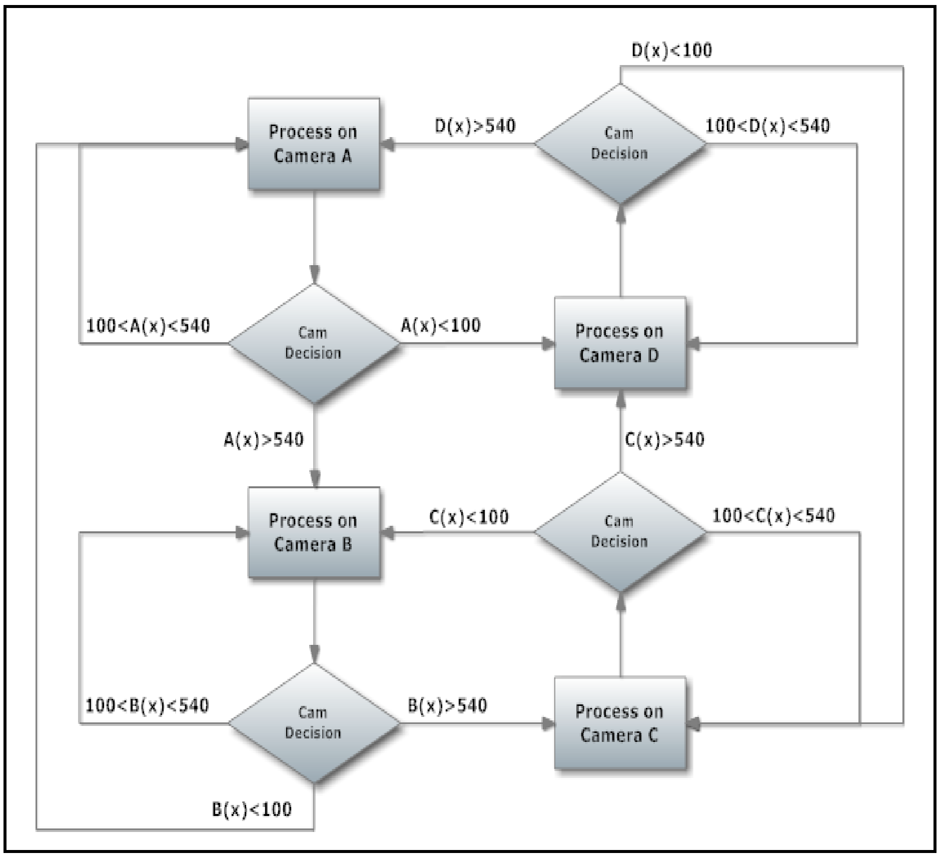

Figure 3.21. Flow Diagram of tracking algorithm for two cameras [70]

There are some extra conditions during tracking introduced below.

- For Example, during the tracking turn of Camera A, if the body turns back and Camera A cannot handle the face, algorithm starts to look for the face in each other cameras, and as it is expected, Camera C could detect the face. Then Algorithm continues to track the face with Camera C. This exemption is also valid for the other situation. If the face turns completely right, then Camera D should find and track the face. If it turns left, Camera B should continue to track.
- If the face is completely invisible in the tracked area or out-of-sight of each camera, system starts to search the face in each camera until the face is found by one of the cameras, and it continues to track with this camera.

4. EXPERIMENTAL RESULTS

4.1. EXPERIMENTS WITH TWO CAMERAS SETUP

Figure 4.1 shows the working of the system with two cameras. In this Figure, we can see the tracking process of the face. It goes from Camera A to B.

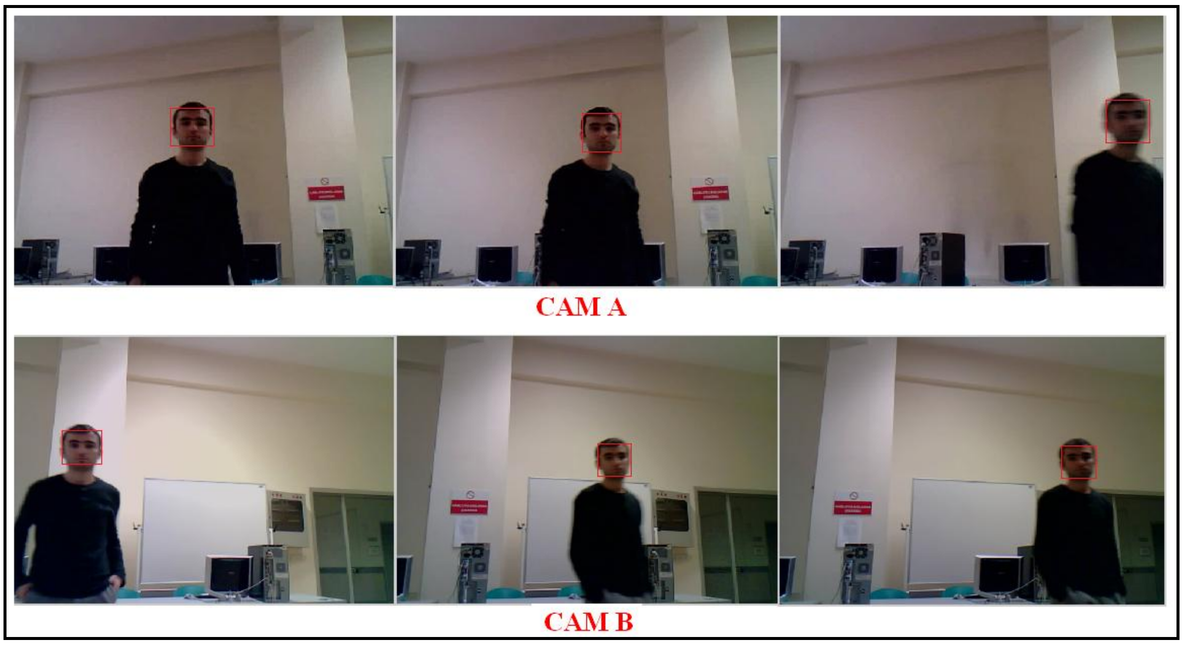

Figure 4.1. Working system with two cameras

4.2. EXPERIMENTS WITH FOUR CAMERAS SETUP

Figure 4.2 shows the images gathered during the run-time of the four camera system. In this figure, tracking is performed sequentially in each camera.

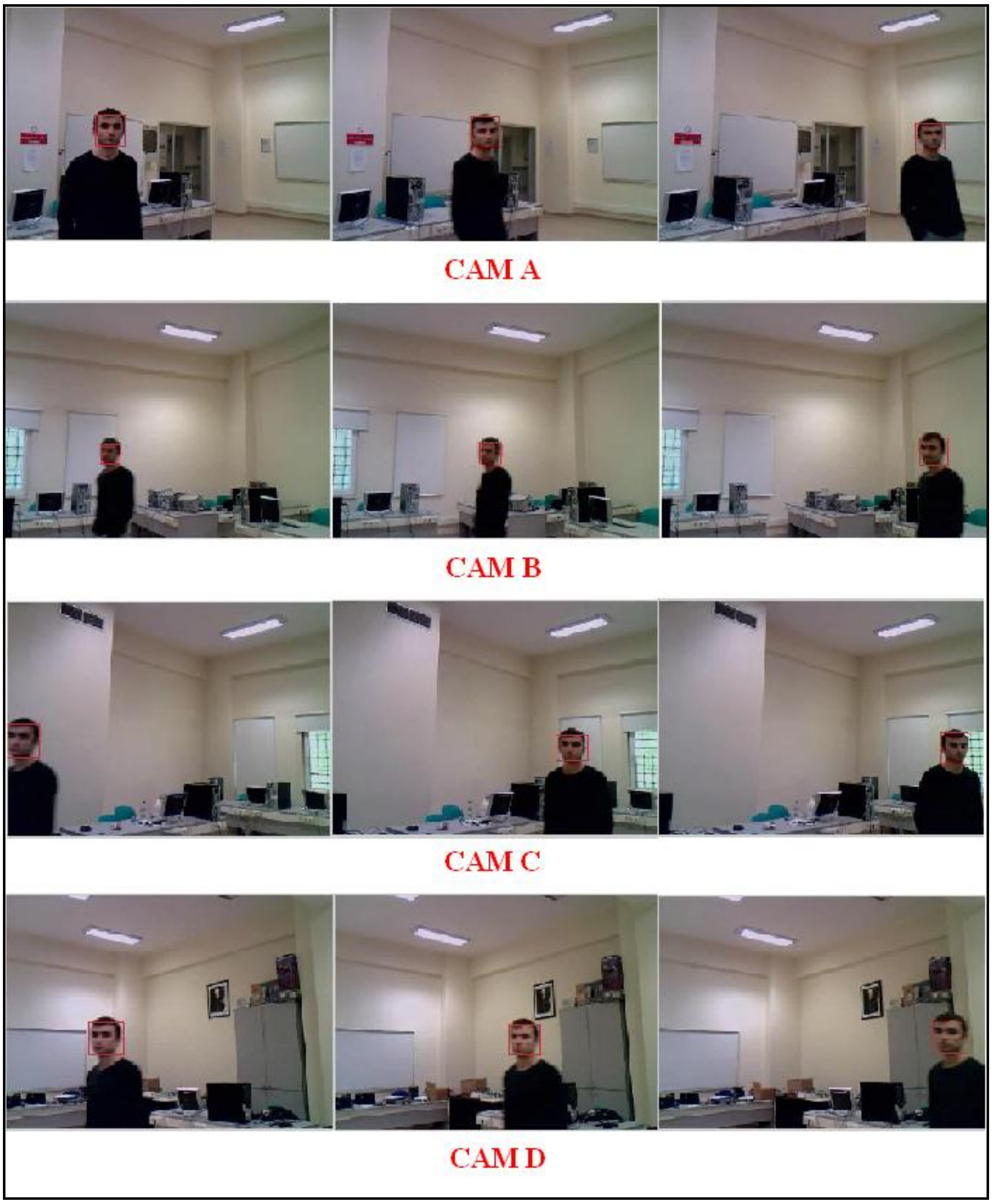

Figure 4.2. Working system with four cameras

4.3. EXPERIMENTS ON FACE RECOGNITION

According to the tests on database images gathered from Face Recognition Homepage [71], we achieved a 80% of success rate after the recognition process. These database images are not used for our complete system. We have tried the actual success rate of the PCA algorithm with the use of these official database images. Figure 4.3 shows some results of the PCA algorithm with using the training database gathered from Face Recognition Homepage [71].

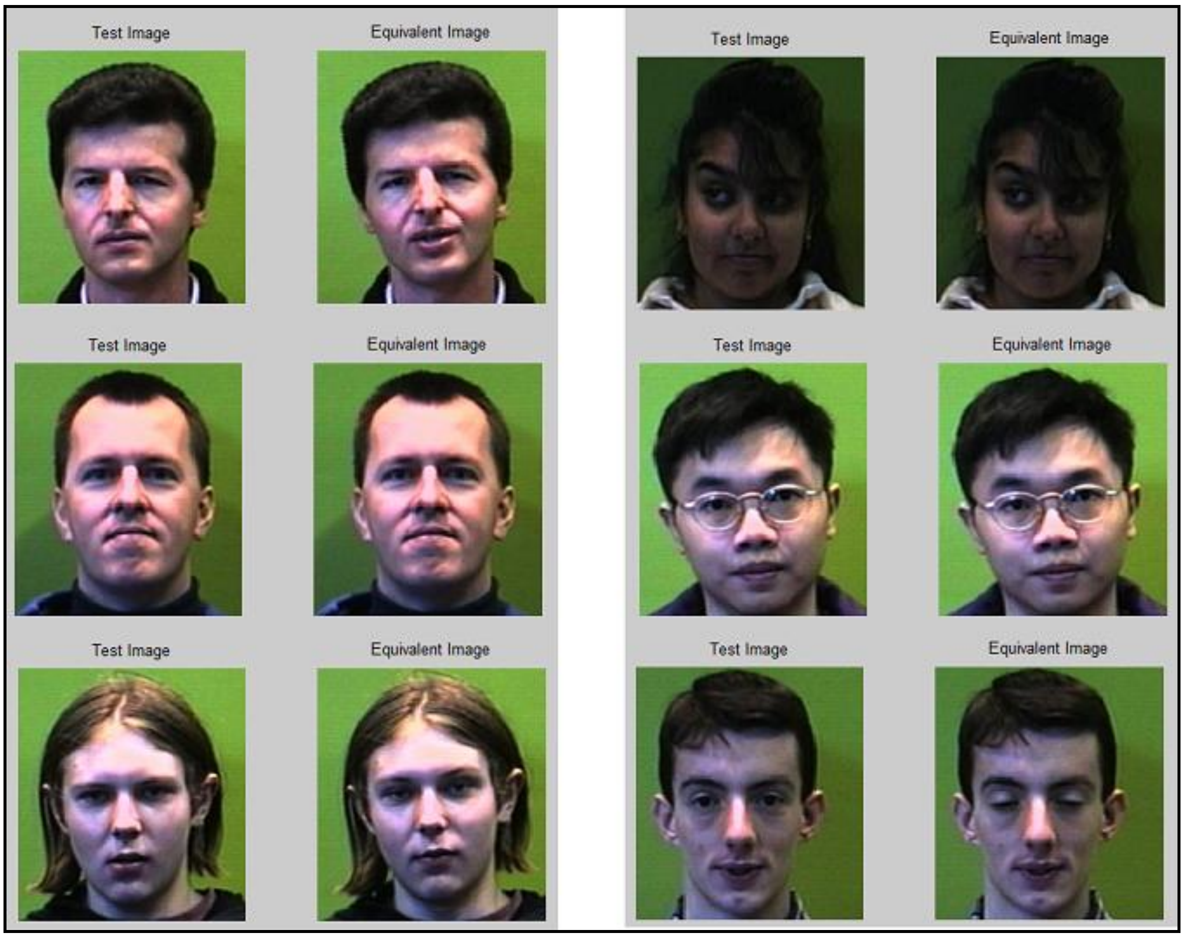

Figure 4.3. Face recognition results of our algorithm using the database in Face Recognition Homepage

In our system, we collected some images of our friends and proposed a training database with these images. Figure 4.4 shows the database images which we use in our system.

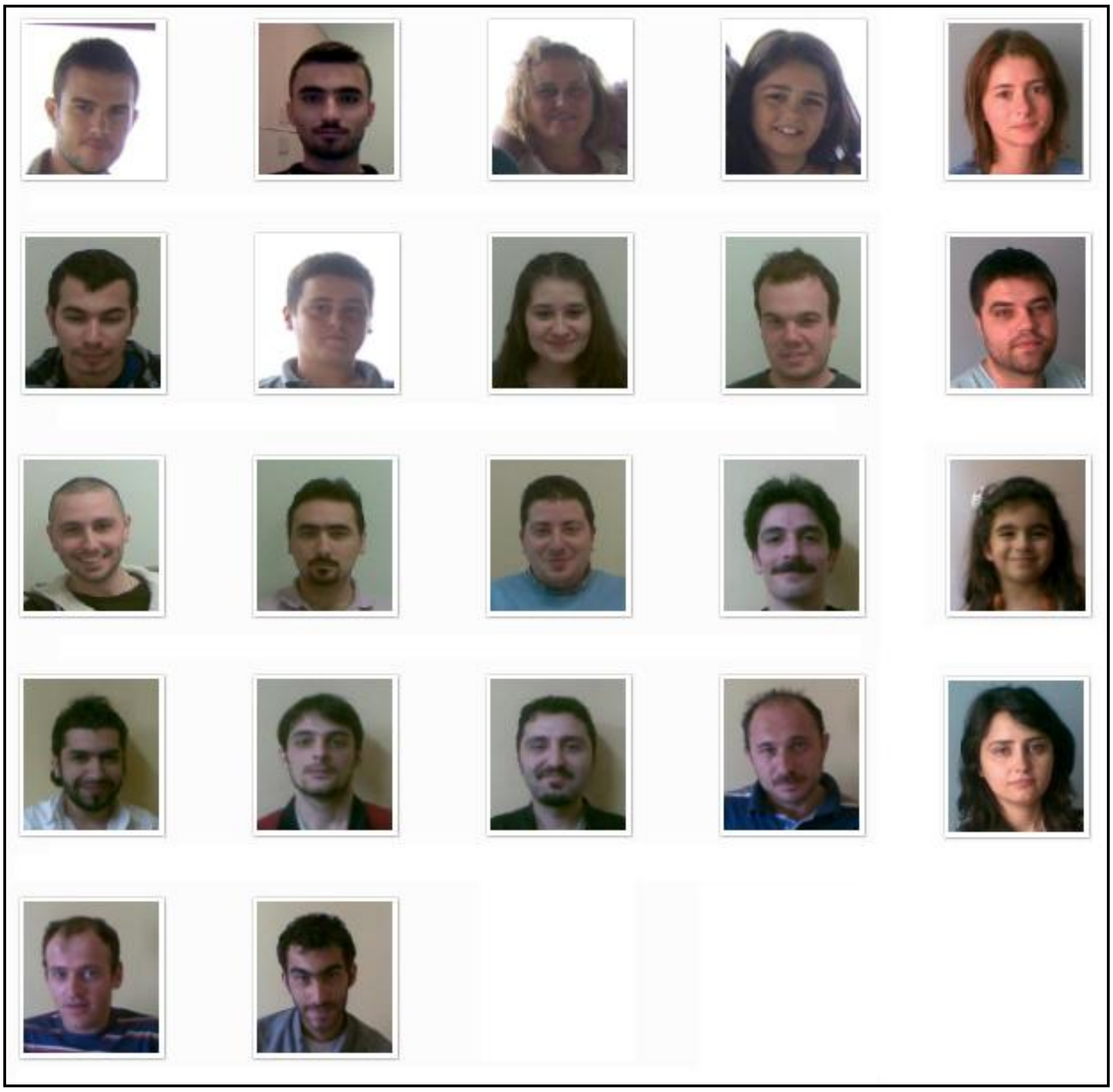

Figure 4.4. Database images used in the system

In our system, as it is explained before, if the human body comes to near the door, we try to recognize its face. After the face is cropped from the input image, we set it as test image and PCA algorithm is applied on to this test image in order to match it with the equivalent image in database. If PCA algorithm cannot recognize the face, we store this face image taken from the system into a null class directory. Result after PCA application is shown in Figure 4.5.

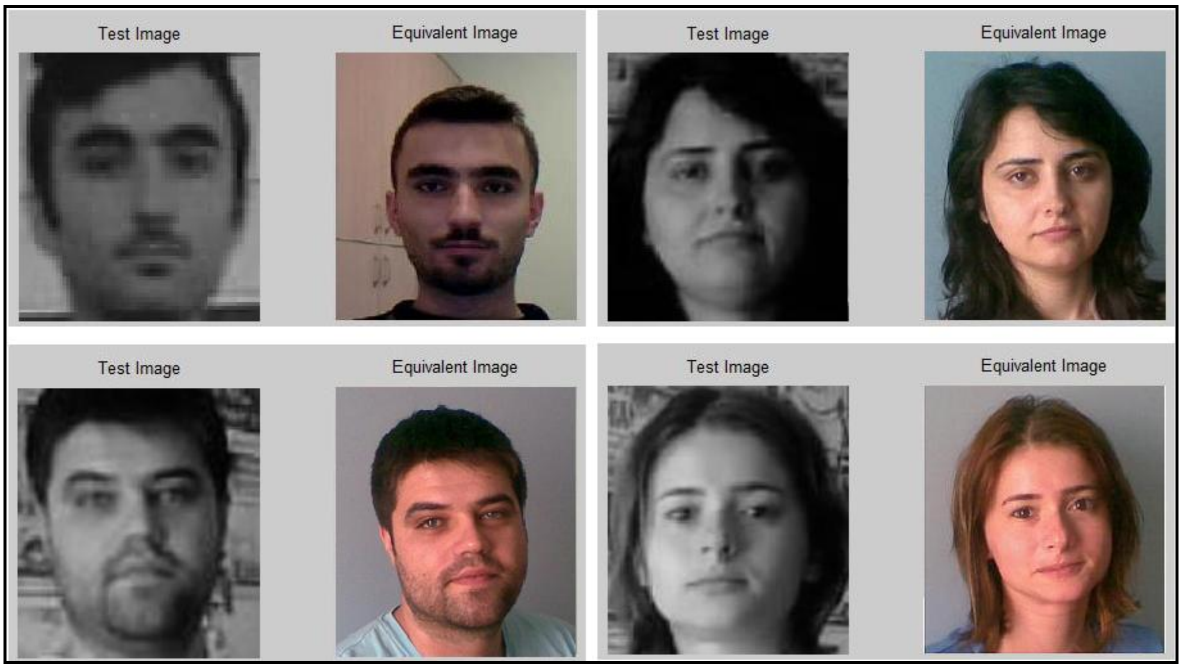

Figure 4.5. Result after the recognition process

4.3. LIMITATIONS OF THE PROPOSED SYSTEMS

Human surveillance systems are vulnerable to development. Our system can also be improved for all sides. Taking video sequences in Matlab was a hard problem. With the improvement in Matlab, this problem can be disregarded and it will be more easy to use Matlab in video surveillance systems.

Using two webcams and integrated them to Matlab are handled easily. However, we have encountered with a problem during integration the other two Sony CCD Color Cameras into the system. This Sony Cameras are used with two TV Cards together in order to receive images. Installing them to Computer and getting images from them was not so easy. We have overcome this issue somehow. It was a driver conflict between the TV Cards. If this hardware problem was resolved by the manufacturer, it will be easier to implement such cameras into our system.

Webcams and Sony Cameras are sufficient in our implementation and the system execution. The resolution for the webcams is 480x640, and for the Sony Cameras is 576x720. When a more high resolution camera is used during the face recognition process, system can recognize the face more successfully.

In future work, this system can be developed in outside areas which have variable light conditions, not stable background images.

Our objective to develop this system was detecting and tracking the one person in indoor area, but our code implementation in face detection is written to detect all the face images. Also we have written the tracking algorithm to track all the faces. However, we tested our tracking algorithm for one person in the tracked area. Tracking more than one person in the area can be developed as a future work.

Another objective was to match and recognizing the face. However, we improved the idea a little bit and put the unrecognized faces into a null class. As a future work, this face images in null class can be integrated to the used database, and we would have achieved a self-improved database.

5. CONCLUSIONS

Face detection technology has come a long way in the last thirty years. Because of the importance of human surveillance in today's world, application of face detection and recognition to human surveillance systems becomes also very important for this reason. Due to these reasons, we considered a surveillance system in this thesis.

Surveillance applications usually work in controlled environments, and detection algorithms can take advantage of the environmental constraints to obtain high detection accuracy. Although detection accuracy is very important in face detection, detection rate and speed is also very important if there will be a human tracking in real time. Due to ease of implementation and having very high detection speed of Adaboost and skin detection algorithms, we used these two algorithms together in our system in order to detect the face.

Tracking of people is the main purpose of the human surveillance systems. Therefore, many tracking algorithms have been developed. We have described the tracking algorithms in Chapter 2. We choose the point tracking, because it is very fast and sufficient for our system to work. We tracked the face within the human tracking system and used the centroid of the face region during tracking which is obtained after the face detection step. This tracking instance is performed first with two cameras and then with four cameras which are settled on the corners of an indoor place.

Tracking of a human is performed of course for a purpose. This is recognition this human. Therefore, we have implemented the face recognition algorithm of PCA to the system in order to recognize the face. Finally, there is still more work to be done, and we believe that more robust face detection and recognition algorithms should be developed.

APPENDIX A: INITIALIZATION FOR RECOGNITION

```
%% Initialization For Recognition

% Read the database directory
Train_images =
dir(fullfile(matlabroot,'work/PCAmelih_opencv/database1/*.jpg'
));

%%%%%%%%%%%%%%%%%%%%%%%% Construction of 2D matrix from 1D
image vectors
T = [];
for i=1:size(Train_images,1)

    % I have chosen the name of each image in databases as a
corresponding number.
    str = int2str(i);
    str = strcat('\',str,'.jpg');
    str =
strcat('F:\MATLAB\R2008a\work\PCAmelih_opencv\database1',str);

    img = imread(str);
    % Reshaping 2D images into 1D image vectors
    img = (rgb2gray(img))';
    temp = img(:);

    % 'T' grows after each turn
    T = [T temp];
end

%%%%%%%%%%%%%%%%%%%%%%%% Calculating the mean image
% Computing the average face image m = (1/P)*sum(Tj's)    (j =
1 : P)
m = mean(T,2);
Train_Number = size(T,2);

%%%%%%%%%%%%%%%%%%%%%%%% Calculating the deviation of each
image from mean image
A = [];
for i = 1 : Train_Number
    % Computing the difference image for each image in the
training set Ai = Ti - m
    temp = double(T(:,i)) - m;
    % Merging all centered images
    A = [A temp];
end

% L is the surrogate of covariance matrix C=A*A'.
L = A'*A;
% Diagonal elements of D are the eigenvalues for both L=A'*A
and C=A*A'.
[V D] = eig(L);

%%%%%%%%%%%%%%%%%%%%%%%% Sorting and eliminating eigenvalues
```

```
% All eigenvalues of matrix L are sorted and those who are
less than a
% specified threshold, are eliminated. So the number of non-
zero
% eigenvectors may be less than (P-1).
L_eig_vec = [];
for i = 1 : size(V,2)
    if( D(i,i)>1 )
        L_eig_vec = [L_eig_vec V(:,i)];
    end
end
%%%%%%%%%%%%%%%%%%%%%%%% Calculating the eigenvectors of
covariance matrix 'C'
% Eigenvectors of covariance matrix C (or so-called
"Eigenfaces")
% can be recovered from L's eigenvectors.
Eigenfaces = A * L_eig_vec;

%%%%%%%%%%%%%%%%%%%%%%%% Projecting centered image vectors
into facespace
% All centered images are projected into facespace by
multiplying in
% Eigenface basis's. Projected vector of each face will be its
corresponding
% feature vector.

ProjectedImages = [];
Train_Number = size(Eigenfaces,2);
for i = 1 : Train_Number
    temp = Eigenfaces'*A(:,i); % Projection of centered images
into facespace
    ProjectedImages = [ProjectedImages temp];
End
```

APPENDIX B: BACKGROUND EXTRACTION

Algorithm B.1. For two cameras

```
%%
vid=videoinput('winvideo',1);
sr=get(vid,'source');
set(sr,'Brightness',100);

set(vid,'FramesPerTrigger',50);
triggerconfig(vid,'manual');
start(vid);
trigger(vid);
img=getdata(vid);
mov=immovie(img);
movie2avi(mov,'background_ev')

%%
vid=videoinput('winvideo',2);
sr=get(vid,'source');
set(sr,'Brightness',60);

set(vid,'FramesPerTrigger',50);
triggerconfig(vid,'manual');
start(vid);
trigger(vid);
img=getdata(vid);
mov=immovie(img);
movie2avi(mov,'background_ev2')
%%
%(1) background with median filter  (1. camera)
med1=aviread('background_ev.avi');
for i=1:size(med1,2)
    med11=med1(i).cdata;
    med12(:,:,i)=rgb2gray(med11);
end
background1=median(med12(:,:,:),3);
%%
%(2) background with median filter  (2. camera)
med2=aviread('background_ev2.avi');
for i=1:size(med2,2)
    med21=med2(i).cdata;
    med22(:,:,i)=rgb2gray(med21);
end
background2=median(med22(:,:,:),3);
```

Algorithm B.2. For four cameras

```
%%%%%%%%%% 1. Camera (Laptop Webcam)
vid=videoinput('winvideo',1);
sr=get(vid,'source');
set(sr,'Brightness',100);
% set(sr,'Contrast',110);
% set(sr,'Exposure',90);
% set(sr,'Gamma',50);
% set(sr,'Hue',180);
set(vid,'FramesPerTrigger',50);
triggerconfig(vid,'manual');

start(vid);
trigger(vid);
img=getdata(vid);

mov=immovie(img);
movie2avi(mov,'background_1')

%%
%%%%%%%%%% 2. camera (External Webcam)
vid=videoinput('winvideo',2);
sr=get(vid,'source');
set(sr,'Brightness',100);
% set(sr,'Contrast',110);
% set(sr,'Exposure',90);
% set(sr,'Gamma',50);
% set(sr,'Hue',180);
set(vid,'FramesPerTrigger',50);
triggerconfig(vid,'manual');

start(vid);
trigger(vid);
img=getdata(vid);

mov=immovie(img);
movie2avi(mov,'background_2')
%%
%%%%%%%%%% 3. camera (Composit Camera 1)
vid=videoinput('winvideo',3);
set(vid,'ReturnedColorSpace','rgb')
triggerconfig(vid, 'manual')
set(vid,'FramesPerTrigger',50)
start(vid)
trigger(vid);
img=getdata(vid);

mov=immovie(img);
movie2avi(mov,'background_3')
%%
%%%%%%%%%% 4. camera (Composit Camera 2)
vid=videoinput('winvideo',4);
set(vid,'ReturnedColorSpace','rgb')
triggerconfig(vid, 'manual')
set(vid,'FramesPerTrigger',50);
```

```
start(vid);
trigger(vid);
img=getdata(vid);

mov=immovie(img);
movie2avi(mov,'background_4')



%%
%(1) background with median filter  (1. camera)
med1=aviread('background_okul1.avi');
for i=1:size(med1,2)
    med11=med1(i).cdata;
    med12(:,:,i)=rgb2gray(med11);
end
background1=median(med12(:,:,:),3);

%(2) background with median filter  (2. camera)
med2=aviread('background_okul2.avi');
for i=1:size(med2,2)
    med21=med2(i).cdata;
    med22(:,:,i)=rgb2gray(med21);
end
background2=median(med22(:,:,:),3);

%(3) background with median filter  (3. camera)
med3=aviread('background_okul3.avi');
for i=1:size(med3,2)
    med31=med3(i).cdata;
    med32(:,:,i)=rgb2gray(med31);
end
background3=median(med32(:,:,:),3);

%(4) background with median filter  (4. camera)
med4=aviread('background_okul4.avi');
for i=1:size(med4,2)
    med41=med4(i).cdata;
    med42(:,:,i)=rgb2gray(med41);
end
background4=median(med42(:,:,:),3);
```

APPENDIX C: FUNCTIONS

Algorithm C.1. Function of Adaboost

```
% Adaboost Face

% Convert the image to grayscale
imgGray = rgb2gray(img);

% Find the faces and draw them
rectangleMatrix = face_detect( imgGray,
classifierFileFullPath, minFaceSize, shouldViewElapsedTime );
```

Algorithm C.2. Function of face_detect

```
function rectangleMatrix = face_detect( img,
classifierFileFullPath, minFaceSize, shouldViewElapsedTime )

if nargin < 3
    minFaceSize = 100;
end
if nargin < 4
    shouldViewElapsedTime = 0;
end

[rectanglesArray] = cvlib_mex('facedetect', im2uint8(img), ...
                              classifierFileFullPath,
minFaceSize, ...
                              shouldViewElapsedTime);
nRectangles = length(rectanglesArray)/4;
rectangleMatrix = zeros(nRectangles,4);
for iRectangle = 1:nRectangles
    rectangleMatrix(iRectangle,:) = ...
        rectanglesArray(4*(iRectangle-1)+1:4*iRectangle);
end
```

Algorithm C.3. Function of Skin_face

```
% Compare the found faces with the skin region

img1=rgb2ycbcr(img);
img2=img1(:,:,3);
img_meancr=mean(img2(:));
Face_coordinates=[];
Face_origin=[];
```

```
for a=1:size(rectangleMatrix,1)
    row=rectangleMatrix(a,:);
    x1=round(row(1,1));
    y1=round(row(1,2));
    x2=round(x1+row(1,3));
    y2=round(y1+row(1,4));
    rectangled_face=img2(y1:y2,x1:x2);
    % Thresholding
    rectangled_face_cr=rectangled_face>(img_meancr+5);
    % Set the centroid of the face image to midpoint
    midpoint=[((x1+x2)/2) ((y1+y2)/2)];

bw=bwareaopen(rectangled_face_cr,round(size(rectangled_face_cr
,1)*size(rectangled_face_cr,2)/2));

    if mean(bw(:))>0

Face_coordinates=[Face_coordinates;rectangleMatrix(a,:)];
        Face_origin=[Face_origin;midpoint];
    end
end

for iRectangle = 1:size(Face_coordinates,1)
    rectangle('Position',Face_coordinates(iRectangle,:),
'EdgeColor', 'r')
end
```

Algorithm C.4. Function of Recognition_face

```
% Recognition Face

for ii=1:size(Face_coordinates,1)
    row=Face_coordinates(ii,:);
    x1=round(row(1,1)-20);
    y1=round(row(1,2)-25);
    x2=round(x1+row(1,3)+40);
    y2=round(y1+row(1,4)+50);
    % if a corner of face image is found out of range of the
image, equal it to '1'
    if x1<1
        x1=1;
    end
    if y1<1
        y1=1;
    end
    if x2<1
        x2=1;
    end
    if y2<1
        y2=1;
    end



    img_red=img(:,:,1);
    cropped_face=img_red(y1:y2,x1:x2);
    % Resizing the face image as a standart 180x200
    cropped_face_2=imresize(cropped_face,[200 180]);


    % Reshaping 2D image into 1D image vector
    [irow icol] = size(cropped_face_2);
    InImage = reshape(cropped_face_2',irow*icol,1);
    % Centered test image
    Difference = double(InImage) - m;
    % Test image feature vector
    ProjectedTestImage = Eigenfaces'*Difference;

    %%%%%%%%%%%%%%%%%%%%%%%% Calculating Euclidean distances
    % Euclidean distances between the projected test image and
the projection
    % of all centered training images are calculated. Test
image is
    % supposed to have minimum distance with its corresponding
image in the
    % training database.

    Euc_dist = [];
    for iii = 1 : Train_Number
        q = ProjectedImages(:,iii);
        temp = ( norm( ProjectedTestImage - q ) )^2;
        Euc_dist = [Euc_dist temp];
    end

    [Euc_dist_min , Recognized_index] = min(Euc_dist);

    % Check if recognized person is not the same with previous
one, because
    % we do not want to recognize the same person
continuously.
    if Recognized_index ~= dummy
        if Euc_dist_min < 3e15
            OutputName =
strcat(int2str(Recognized_index),'.jpg');

            SelectedImage =
strcat('F:\MATLAB\R2008a\work\PCAmelih_opencv\database1','\',O
utputName);
            SelectedImage = imread(SelectedImage);
            close all
            figure(2);subplot(1,2,1);imshow(cropped_face_2)

            title('Test Image');
            figure(2);subplot(1,2,2);imshow(SelectedImage);
            title('Equivalent Image');
        end
        if  Euc_dist_min >= 3e15
            fprintf('no match found')
            str = int2str(null);
            str = strcat('\',str,'.jpg');
            str =
strcat('F:\MATLAB\R2008a\work\PCAmelih_opencv\nullclass',str);
            imwrite(cropped_face_2,str);
            null=null+1;
        end
        dummy = Recognized_index;
    end
end
```

Algorithm C.5. Function of camera3_set

```
function vid3=camera3_set()
vid3=videoinput('winvideo',3);
set(vid3,'ReturnedColorSpace','rgb')
triggerconfig(vid3, 'manual')
set(vid3,'FramesPerTrigger',1)
start(vid3)
```

Algorithm C.6. Function of camera4_set

```
function vid4=camera4_set()
vid4=videoinput('winvideo',4);
set(vid4,'ReturnedColorSpace','rgb')
triggerconfig(vid4, 'manual')
set(vid4,'FramesPerTrigger',1)
start(vid4)
```

Algorithm C.7. Function of camera3_get

```
function img=camera3_get(vid3)
trigger(vid3);
img=getdata(vid3);
```

Algorithm C.8. Function of camera4_get

```
function img=camera4_get(vid4)
trigger(vid4);
img=getdata(vid4);
```

APPENDIX D: MAIN FUNCTIONS

Algorithm D.1. For two cameras

```
%% real time face tracking

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialization
of Cameras
vid1=videoinput('winvideo',1);
vid2=videoinput('winvideo',2);
sr1=get(vid1,'source');
set(sr1,'Brightness',100);
sr2=get(vid2,'source');
set(sr2,'Brightness',60);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialization
for Adaboost algorithm
classifierFileFullPath =
['F:\MATLAB\R2008a\work\PCAmelih_opencv\haarcascades\haarcasca
de_frontalface_default.xml'];
% Minimum face size (area) to detect
minFaceSize = 15;
% Should the OpenCV face detector print the elapsed time?
shouldViewElapsedTime = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% null variable is used to write the unrecognized images into
nullclass file
null=1;
% SE variable is used for imdilation operation in Skin_face
SE = strel('disk',5);

dummy=[];
cam=1;
tr=2;
background1=rgb2gray(background1);
background2=rgb2gray(background2);
while (1)

    if cam==1
        img=getsnapshot(vid1);
        imgGray=rgb2gray(img);
        subtract_img1=abs(imgGray-background1);
        bw1=bwareaopen((subtract_img1<5),100);

    else
        img=getsnapshot(vid2);
        imgGray=rgb2gray(img);
        subtract_img2=abs(imgGray-background2);
        bw1=bwareaopen((subtract_img2<5),100);

    end
        imshow(img,[])
    if (mean(bw1(:))> 0)
        Adaboost_face
        if (size(rectangleMatrix,1)>0)
            Skin_face
            if (size(Face_coordinates,1)> 0)

                if (cam==1 && Face_origin(1,1)<500)
                    tr=1;
                end

                if (cam==1 && Face_origin(1,1)>500)
                    tr=2;
                end

                if (cam==2 && Face_origin(1,1)>100)
                    tr=2;
                end

            end
        end

    end


    if tr==2
        cam=2;
        tr=1;
    elseif tr==1
        cam=1;
        tr=2;
    end


    if (cam==2 && Face_origin(1,1)>500 && row(1,3)>150)
        Recognition_face
    end
end
```

Algorithm D.2. For four cameras

```
%% real time face tracking

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialization
of Cameras
vid1=videoinput('winvideo',1);
vid2=videoinput('winvideo',2);
camera3_set;
camera4_set;
sr1=get(vid1,'source');
set(sr1,'Brightness',100);
sr2=get(vid2,'source');
```

```
set(sr2,'Brightness',60);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Initialization
for Adaboost algorithm
classifierFileFullPath =
['F:\MATLAB\R2008a\work\PCAmelih_opencv\haarcascades\haarcasca
de_frontalface_default.xml'];
% Minimum face size (area) to detect
minFaceSize = 15;
% Should the OpenCV face detector print the elapsed time?
shouldViewElapsedTime = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% null variable is used to write the unrecognized images into
nullclass file
null=1;
% SE variable is used for imdilation operation in Skin_face
SE = strel('disk',5);

dummy=[];
cam=1;
tr=4;
background1=rgb2gray(background1);
background2=rgb2gray(background2);
background3=rgb2gray(background3);
background4=rgb2gray(background4);
while (1)

    if cam==1
        img=getsnapshot(vid1);
        img10=rgb2gray(img);
        subtract_img1=abs(img10-background1);
        bw1=bwareaopen((subtract_img1<5),100);
    elseif cam==2
        img=getsnapshot(vid2);
        img10=rgb2gray(img);
        subtract_img2=abs(img10-background2);
        bw1=bwareaopen((subtract_img2<5),100);
    elseif cam==3
        img=camera3_get(vid3);
        img10=rgb2gray(img);
        subtract_img3=abs(img10-background3);
        bw1=bwareaopen((subtract_img3<5),100);
    else
        img=camera4_get(vid4);
        img10=rgb2gray(img);
        subtract_img4=abs(img10-background4);
        bw1=bwareaopen((subtract_img4<5),100);

    end
    imshow(img,[])



    if (mean(bw1(:))> 0)
        Adaboost_face
        if (size(rectangleMatrix,1)>0)
            Skin_face
            if (size(Face_coordinates,1)> 0)
```

```
                if (cam==1 && 100<Face_origin(1,1)<500)
                    tr=3;
                elseif (cam==1 && Face_origin(1,1)>500)
                    tr=4;
                elseif (cam==1 && Face_origin(1,1)<100)
                    tr=2;
                elseif (cam==2 && Face_origin(1,1)<100)
                    tr=3;
                elseif (cam==2 && 100<Face_origin(1,1)<500)
                    tr=4;
                elseif (cam==2 && Face_origin(1,1)>500)
                    tr=1;
                elseif (cam==3 && Face_origin(1,1)<100)
                    tr=4;
                elseif (cam==3 && 100<Face_origin(1,1)<500)
                    tr=1;
                elseif (cam==3 && Face_origin(1,1)>500)
                    tr=2;
                elseif (cam==4 && Face_origin(1,1)<100)
                    tr=1;
                elseif (cam==4 && 100<Face_origin(1,1)<500)
                    tr=2;
                elseif (cam==4 && Face_origin(1,1)>500)
                    tr=3;
                end

            end
        end

    end


    if tr==4
        cam=2;
        tr=1;
    elseif tr==3
        cam=1;
        tr=4;
    elseif tr==2
        cam=4;
        tr=3;
    elseif tr==1
        cam=3;
        tr=2;
    end

    if (cam==2 && Face_origin(1,1)>500 && row(1,3)>150)
        Recognition_face
    end
end
```

REFERENCES

1. Kotropoulos, C. and I., Pitas, “Rule-Based Face Detection in Frontal Views”, Proc. Int’l Conf. Acoustics, *Speech and Signal Processing*, vol. 4, pp. 2537-2540, 1997.

2. Yang, G. and T.S., Huang, “Human Face Detection in Complex Background”, *Pattern Recognition*, vol. 27, no. 1, pp. 53-63, 1994.

3. Sirohey, S.A., *Human Face Segmentation and Identification*, Technical Report CS-TR-3176, Univ. of Maryland, 1993.

4. Graf, H.P., T., Chen, E., Petajan and E., Cosatto, “Locating Faces and Facial Parts”, Proc. First Int’l Workshop, *Automatic Face and Gesture Recognition*, pp. 41-46, 1995.

5. Leung, T.K., M.C., Burl and P., Perona, “Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching”, Proc. Fifth IEEE Int’l Conf. *Computer Vision*, pp. 637-644, 1995.

6. Burl, M.C., T.K., Leung and P., Perona, “Face Localization via Shape Statistics”, Proc. First Int’l Workshop, *Automatic Face and Gesture Recognition*, pp. 154-159, 1995.

7. Ghorbel, M.B., M., Baklouti and S., Couvet, “3D Head Pose Estimation and Tracking Using Particle Filtering and ICP Algorithm”, *Lecture Notes in Computer Science*, 2010.

8. Viola, P. and M.J., Jones, *Robust Real-Time Face Detection*, International Journal of Computer Vision 57(2), 2004.

9. Jebara, T.S. and A., Pentland, “Parameterized Structure from Motion for 3D Adaptive Feedback Tracking of Faces”, Proc. IEEE Conf., *Computer Vision and Pattern Recognition*, pp. 144-150, 1997.

10. Yang, J. and A., Waibel, “A Real-Time Face Tracker”, Proc. Third Workshop, *Applications of Computer Vision*, pp. 142-147, 1996.

11. Crowley, J.L. and F., Berard, “Multi-Modal Tracking of Faces for Video Communications”, Proc. IEEE Conf., *Computer Vision and Pattern Recognition*, pp. 640-645, 1997.

12. Kim, S.H., N.K., Kim, S.C., Ahn and H.G., Kim, “Object Oriented Face Detection Using Range and Color Information”, Proc. Third Int’l Conf., *Automatic Face and Gesture Recognition*, pp. 76-81, 1998.

13. Chai, D. and K.N., Ngan, “Locating Facial Region of a Head-and-Shoulders Color Image”, Proc. Third Int’l Conf., *Automatic Face and Gesture Recognition*, pp. 124-129, 1998.

14. Wang, H. and S. F., Chang, “A Highly Efficient System for Automatic Face Region Detection in MPEG Video”, IEEE Trans., *Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 615-628, 1997.

15. Kjeldsen, R. and J., Kender, “Finding Skin in Color Images”, Proc. Second Int’l Conf., *Automatic Face and Gesture Recognition*, pp. 312-317, 1996.

16. Dai, Y. and Y., Nakano, “Extraction for Facial Images from Complex Background Using Color Information and SGLD Matrices”, Proc. First Int’l Workshop, *Automatic Face and Gesture Recognition*, pp. 238-242, 1995.

17. Saber, E. and A.M., Tekalp, “Frontal-View Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry Based Cost Functions”, *Pattern Recognition Letters*, vol. 17, no. 8, pp. 669-680, 1998.

18. Augusteijn, M.F. and T.L., Skujca, "Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology", Proc. IEEE Conf., *Neural Networks*, pp. 392-398, 1993.

19. Graf, H.P., E., Cosatto, D., Gibbon, M. Kocheisen and E., Petajan, "Multimodal System for Locating Heads and Faces", Proc. Second Int'l Conf., *Automatic Face and Gesture Recognition*, pp. 88-93, 1996.

20. Yang, M.H. and N., Ahuja, "Detecting Human Faces in Color Images", Proc. IEEE Int'l Conf., *Image Processing*, vol. 1, pp. 127-130, 1998.

21. Sakai, T., M., Nagao and S., Fujibayashi, "Line Extraction and Pattern Detection in a Photograph", *Pattern Recognition*, vol. 1, pp. 233-248, 1969.

22. Craw, I., H., Ellis and J., Lishman, "Automatic Extraction of Face Features", *Pattern Recognition Letters*, vol. 5, pp. 183-187, 1987.

23. Govindaraju, V., S.N., Srihari and D.B., Sher, "A Computational Model for Face Location", Proc. Third IEEE Int'l Conf., *Computer Vision*, pp. 718-721, 1990.

24. Hjelmas, R. and B.K., Low, "Face detection: a survey", *Computer Vision Image Understanding*, vol. 83, pp. 236–274, 2001.

25. Rowley, H.A., S., Baluja and T., Kanade, "Neural network-based face detection", IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 20, pp. 23–38, 1998.

26. Sung, K.K. and T., Poggio, "Example-based learning for view-based human face detection", IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 20, pp. 39–50, 1998.

27. Schneiderman, H. and T., Kanade, "Object detection using the statistic of parts", Int. J., *Computer Vision*, vol. 56, pp. 151–177, 2004.

28. Turk, M. and A., Pentland, “Eigenfaces for Recognition”, *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

29. Osuna, E., R., Freund and F., Girosi, “Training support vector machines: an application to face detection”, Proc. IEEE Conf., *Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

30. Sung, K.K. and T., Poggio, “Example-Based Learning for View-Based Human Face Detection”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39-51, Jan. 1998.

31. Yang, M.H., N., Ahuja and D., Kriegman, “Mixtures of Linear Subspaces for Face Detection”, Proc. Fourth Int’l Conf., *Automatic Face and Gesture Recognition*, pp. 70-76, 2000.

32. Gutta, S. and H., Wechsler, “Face Recognition using Asymmetric Faces”, *ICBA.*, pp. 162-168, 2004.

33. Duda, R.o., P.E., Hart and D.G., Stork, *Pattern Classification*, NY: John Wiley & Sons, 2001.

34. Turk, M.A. and A.P., Pentland, ”Face recognition using eigenfaces”, Proc. IEEE Computer Society Conf., *Computer Vision and Pattern Recognition*, pp. 586- 591, Maui, Hawaii, 1991.

35. Liu, N., H., Wang and W.Y., Yau, *Face Recognition with Weighted Kernel Principal Component Analysis*

36. Blanz, V. and T., Vetter, “Face recognition based on fitting a 3D morphable model”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 25, Issue 9, Sept. 2003.

37. Huang J., B., Heisele, and V., Blanz, *Component-based Face Recognition with 3D Morphable Models*

38. Wiskott, L., J.M., Fellous, N., Kruger and Christoph von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching", *Intelligent Biometric Techniques in Fingerprint and Face Recognition*, Chapter 11, pp. 355-396, 1999.

39. Achermann, B. and H., Bunke, "Combination of Face Classifiers for Person Identification", Proc. 13th IAPR International Conference, *Pattern Recognition (ICPR),* vol. 3, pp. 416-420, Vienna, 1996.

40. Samaria, F. and S., Young, "HMM-Based Architecture for Face Identification", *Image and Vision Computing*, vol. 12, Issue 8, pp. 537-543, 1994.

41. Cootes, T., T., Taylor, C., Cooper and D., Graham, "Active Shape Models – Their Training and Application", *Computer Vision and Image Understanding*, pp. 38-59, 1995.

42. Comaniciu, D., V., Ramesh and P., Meer, "Kernel-based object tracking", IEEE Trans, *Pattern Analysis and Machine Intelligence,* vol. 25, pp. 564–575, 2003.

43. Schweitzer, H., J.W., Bell and F., Wu, "Very fast template matching", European Conf., *Computer Vision (ECCV),* pp. 358–372, 2002.

44. Fieguth, P. and D., Terzopoulos, "Color-based tracking of heads and other mobile objects at video frame rates", IEEE Conf., *Computer Vision and Pattern Recognition (CVPR),* pp. 21–27, 1997.

45. Jepson, A., D., Fleet and T., Elmaraghi, "Robust online appearance models for visual tracking", IEEE Trans., *Pattern Analysis and Machine Intelligence,* vol. 25, Issue 10, pp. 1296–1311, 2003.

46. Shi, J. and C., Tomasi, "Good features to track", IEEE Conf., *Computer Vision and Pattern Recognition (CVPR),* pp. 593–600, 1994.

47. Tao, H., H., Sawhney and R., Kumar, “Object tracking with Bayesian estimation of dynamic layer representations”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 24, Issue 1, pp. 75–89, 2002.

48. Isard, M. and J., Maccormick, “Bramble: A bayesian multiple-blob tracker”, IEEE Int. Conf., *Computer Vision (ICCV).* 34–41, 2001.

49. Black, M. and A., Jepson, “Eigentracking: Robust matching and tracking of articulated objects using a view-based representation”, *Int. J. Computer Vision,* vol. 26, Issue 1, pp. 63–84, 1998.

50. Avidan, S., “Support vector tracking”, IEEE Conf., *Computer Vision and Pattern Recognition (CVPR),* pp. 184–191, 2001.

51. Haritaoglu, I., D., Harwood and L.S., Davis, “Hydra: multiple people detection and tracking using silhouettes”, Second IEEE Workshop, *Visual Surveillance (VS'99),* 1999.

52. Kang, J., I., Cohen and G., Medioni, “Object reacquisition using geometric invariant appearance model”, Int. Conf., *Pattern Recongnition (ICPR),* pp. 759–762, 2004.

53. Sato, K. and J., Aggarwal, “Temporal spatio-velocity transform and its application to tracking and interaction”, *Computer Vision Image Understand,* vol. 96, Issue 2, pp. 100–128, 2004.

54. Terzopoulos, D. and R., Szeliski, “Tracking with kalman snakes”, *Active Vision,* A. Blake and A. Yuille, *Eds. MIT Press,* 1992.

55. Maccormick, J. and A., Blake, “Probabilistic exclusion and partitioned sampling for multiple object tracking”, Int. J., *Computer Vision*, vol. 39, Issue 1, pp. 57–71, 2000.

56. Chen, Y., Y., Rui and T., Huang, “Jpdaf based hmm for real-time contour tracking”, IEEE Conf., *Computer Vision and Pattern Recognition (CVPR),* pp. 543–550, 2001.

57. Bertalmio, M., G., Sapiro and G., Randall, “Morphing active contours”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 22, Issue 7, pp. 733–737, 2000.

58. Mansouri, A., “Region tracking via level set pdes without motion computation”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 24, Issue 7, pp. 947–961, 2002.

59. Ronfard, R., “Region based strategies for active contour models”, Int. J., *Computer Vision*, vol. 13, Issue 2, pp. 229–251, 1994.

60. Yılmaz, A., X., Li and M., Shah, “Contour based object tracking with occlusion handling in video acquired using mobile cameras”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 26, Issue 11, pp. 1531–1536, 2004.

61. Arshad, M.R. and N., Nordin, “Human Motion Tracking and Analysis via Point Tracking Technique”, Int. Conf., *Control, Instrumentation and Mechatronics Engineering (CIM '07)*, 2007.

62. Kuhn, H., “The hungarian method for solving the assignment problem”, *Naval Research Logistics Quart.*, vol. 2, pp. 83–97, 1955.

63. Sethi, I. and R., Jain, “Finding trajectories of feature points in a monocular image sequence”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 9, Issue 1, pp. 56–73, 1987.

64. Salari, V. and I.K., Sethi, “Feature point correspondence in the presence of occlusion”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 12, Issue 1, pp. 87–91, 1990.

65. Rangarajan, K. and M., Shah, “Establishing motion correspondence”, Conf., *Vision Graphies Image Process*, vol. 54, Issue 1, pp. 56–73, 1991.

66. Veenman, C., M., Reinders, and E., Backer, “Resolving motion correspondence for densely moving points”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 23, Issue 1, pp. 54–72, 2001.

67. Intille, S., J., Davis and A., Bobick, “Real-time closed-world tracking”, IEEE Conf., *Computer Vision and Pattern Recognition (CVPR)*, pp. 697–703, 1997.

68. Broida, T. and R., Chellappa, “Estimation of object motion parameters from noisy images”, IEEE Trans., *Pattern Analysis and Machine Intelligence*, vol. 8, Issue 1, pp. 90–99, 1986.

69. Cox, I.J., “A review of statistical data association techniques for motion correspondence”, Int. J., *Computer Vision*, vol. 10, Issue 1, pp. 53–66, 1993.

70. Communicate Visually Company, “SmartDraw Software”, Available on site *http://www.smartdraw.com*

71. “Face Recognition Database Images”, Available on site *http://www.face-rec.org/databases*